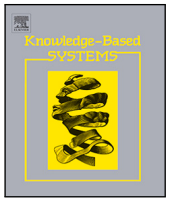
Knowledge-Based Systems 280 (2023) 111042

Contents lists available at ScienceDirect

Knowledge-Based Systems

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



Variational mode decomposition and sample entropy optimization based transformer framework for cloud resource load prediction

Jiaxian Zhu [a], Weihua Bai [a], Jialing Zhao [a], Liyun Zuo [b], Teng Zhou [c,d,*], Keqin Li [e]

[a] School of Computer Science, Zhaoqing University, Zhaoqing 526061, China
[b] School of Computer Science, Guangdong University of Petrochemical Technology, Zhanjiang 525000, China
[c] School of Cyberspace Security, Hainan University, Haikou 570228, China
[d] Centre for Smart Health, The Hong Kong Polytechnic University, Kowloon, Hong Kong
[e] Department of Computer Science, State University of New York, New Paltz, New York 12561, USA

ARTICLE INFO

Keywords:
Cloud resource prediction
Encoding representation
Multidimensional hidden factors
Sample entropy
Variational mode decomposition

ABSTRACT

The efficient prediction of cloud resource demand plays a crucial role in resource allocation and scheduling in cloud data centers, helping to optimize resource utilization and improve service quality. However, accurately predicting cloud resource demand poses challenges due to the failure of prediction models in real-world scenarios, such as extreme load peaks, and the limitation of computation burden on the global characterization capability. To effectively handle single-variable cloud resource load time series with multidimensional hidden factors, we propose a sample entropy-optimized variational model decomposition transformer (VMDSE-Tformer) for cloud resource scheduling. Hereby, we decompose the time series through variational model decomposition, and then reconstruct the subsequence collection using sample entropy calculation. Then, we use a class Transformer framework with a multi-head self-attention mechanism to learn deep features and obtain encoding representations of each component sequence. We conduct sufficient experiments on three benchmark datasets by comparing them with five state-of-the-art models. Notably, the MAPE of VMDSE-Tformer is improved by about 60% compared to LSTNet. The results demonstrate the superior performance of our VMDSE-Tformer in terms of predicting task sequence intensity, CPU, and RAM resource demand. Therefore, VMDSE-Tformer can serve as a powerful and efficient tool to predict resource demand in cloud data centers, with implications for more effective resource management and service delivery.

1. Introduction and motivations

Infrastructure resources such as central processing unit (CPU), memory, network bandwidth, and storage constitute the main components of cloud resources and serve as a crucial support environment for collaborative cloud–edge-end applications. Accurate computation resource load demand prediction is a key technique for intelligent cloud resource control and scheduling in data centers, which plays an important role in the operation and management of task and resource scheduling systems. Predicting and allocating appropriate computing resources for cloud–edge-end collaborative application execution can better support the operation of various services to improve resource utilization, reduce production costs, and save energy while enhancing overall system performance [1–3]. To accurately predict cloud resource demands can better enable pre-allocation and management control of various resources and adapt to load changes in cloud data center systems, which can provide intelligent decision-making references to reduce energy consumption and costs and improve system performance. Management of resource scheduling based on cloud resource load prediction is one of the most strategic and necessary research areas in cloud computing. Inaccurate demand prediction can result in overloading or congestion in cloud resource allocation and scheduling, which will affect task scheduling plans. Based on the task intensity and request data for various resources in a time series, predicting task intensity and cloud resource demands is the core task of load prediction models in cloud data centers, and an indispensable part of intelligent and perception-based cloud resource scheduling systems. The demand for task intensity or cloud resource load is carried by various observable factors (*e.g.*, user type, application service type, price fluctuations in resources or traffic) and invisible factors (*e.g.*, time periods, regions, and application structures) that interact. It forms the basis for integrated task and resource allocation, control, and scheduling. In cloud data centers, the obtained data are present as univariate time series, reflecting the demand for CPU, RAM, storage, and other resources [4–7]. However,

* Corresponding author at: School of Cyberspace Security, Hainan University, Haikou 570228, China.
E-mail addresses: baiweihua@zqu.edu.cn (W. Bai), teng.zhou@hainanu.edu.cn (T. Zhou).

https://doi.org/10.1016/j.knosys.2023.111042
Received 28 July 2023; Received in revised form 25 September 2023; Accepted 28 September 2023
Available online 4 October 2023

this seemingly simple univariate time series is actually the result of the combined effects of various explicit or implicit factors, such as user types, application service types, and price fluctuations, as well as the demands of applications. The key to solving prediction problems lies in exploring the strong coupling relationships among hidden factors within the univariate time series, where the analysis and extraction of features can improve the predictability of cloud resource loads and the accuracy of prediction models.

The analysis of univariate time series to predict cloud resource loads has attracted great interest in the field of cloud computing, and various traditional prediction approaches have been proposed, such as Kalman Filtering (KF), Support Vector Machine (SVM), Autoregressive Integrated Moving Average (ARIMA), Fuzzy Linear Regression, and neural network models such as Long Short-Term Memory (LSTM), Gradient Boosting Regression Trees (GBRTs), Artificial Neural Networks (ANNs), eXtreme Gradient Boosting (XGBoost), and Recurrent Neural Networks (RNNs) [2,3,8,9]. These approaches have limitations and drawbacks when dealing with cloud resource demand loads that exhibit unobservable latent and multidimensional dependent features as well as long-term temporal dependencies.

Recent studies [10–12] propose to optimize the structure of associated datasets to enhance logical mining, using the Hopfield neural network to obtain optimal logical rules from correlations and learn to extract the corresponding rules, and experimentally verifying the effectiveness of mining optimal logical rules to analyze data. However, since no associated univariate time-series dataset is available, it is not feasible to optimize the structure of the associated dataset and mine optimal logical rules. Hence this method is not suitable for analyzing such data.

In cloud data centers, the cloud resource demand lacks support from observable factors and contains multidimensional hidden factors, such as the diversity of users, complexity of applications, network bandwidth fluctuations, and temporal differences, which interact to affect cloud resource demand. To improve the accuracy and reliability of cloud data center load forecasting, we propose VMDSE-Tformer, an attention mechanism framework for resource scheduling prediction. The VMDSE-Tformer leverages the robustness of multiple adaptive Wiener filters to process noisy univariate cloud resource demand sequences and does not require mutually independent latent variables by sample entropy-optimized variational mode decomposition (VMD). Our framework handles the load time series of univariate cloud resource demand that encompasses multidimensional hidden factors. The efficacy of the state uncertainty level of each decomposition component is evaluated through its sample entropy. After selecting the optimal embedding dimension K (the number of modes) and similarity tolerance τ, a sample data structure is constructed for constructing the predictive model. A class of Transformer framework based on Multi-head Self-attention is used to learn the strong spatiotemporal correlation features and internal latent factors from each mode component sequence, which results in a representation code that captures multi-correlation information at each timestamp. After concatenating all the mode features, a neural network with one hidden layer is used for regression learning on the concatenated representation codes to complete cloud resource load forecasting. The cloud resource load prediction model aims to forecast application requests for task flow, CPU, RAM, and other resource scheduling for the next 2–4 h based on a univariate time-tagged historical data segment, *e.g.*, the historical data 3 or 6 h before the prediction time point, by analyzing cloud data. The framework employs a multi-head attention mechanism with multi-hidden-factor feature fusion learning based on sample entropy optimized high-embedding-dimension modal decomposition, and is also suitable for other time series applications such as public health [13] or traffic flow forecasting [14].

This paper makes the following contributions:

- A data processing and prediction model uses multi-head attention to learn hidden features of single-variable time-series through VMD based on sample entropy selection;
- A learning mode focuses on the strong coupling correlation features between multiple independent factors within a single-variable time series and presents a learning framework to fuse strongly coupled correlation features and hidden factor features in the modal component sequence regarding their position or time point;
- Experimental results on a dataset of service task sequences and resource request sequences in cloud data centers validate the performance of the proposed model in handling time-series data on task and cloud resource loads. The proposed model achieves an average improvement of 24.31%, 10.5%, and 9.75% on MAPE, RMSE, and RSE, respectively, compared to Transformer on the Workload dataset. It also outperforms LSTNet [15] with average improvements of 59.65%, 34.8%, and 28.2% by the same metrics, respectively.

The rest of this paper is organized as follows. In Section 2, we briefly review the related works. A VMDSE-Tformer model for cloud resource scheduling is presented in Section 3. In Section 4, we present the parameter selection and optimization techniques for VMD, as well as the reconstruction of subsequences for the model. Sufficient experiments in Section 5 demonstrate the superiority of the proposed method. In Section 6, we conclude and discuss future work.

2. Related work

The analysis of cloud resource usage and the development of load prediction models rely primarily on time-series datasets constructed from a continuous set of data points in cloud data centers, capturing critical metrics such as CPU utilization, RAM usage, and bandwidth applications over time. These datasets form the fundamental basis for this research. Many studies have investigated and developed models for time-series data obtained from resource monitoring in cloud data centers using traditional techniques [8–12,16] such as Kalman Filter (KF), Autoregressive Integrated Moving Average (ARIMA), Random Forest (RF), Support Vector Regression (SVR), eXtreme Gradient Boosting (XGBoost), and Long Short-term Memory (LSTM), to forecast cloud resource loads by analyzing raw time-series data on resource demands and constructing predictive models based on fitting or regressing historical data. AbdElminaam et al. [10] developed an analytics and prediction framework using a quantum genetic algorithm to optimize a Kalman filtering neural fuzzy system to analyze and predict Google's server cluster data. To analyze and forecast Key Performance Indicator (KPI) time-series data of cloud servers, Gyeera et al. constructed a prediction framework utilizing optimized Kalman filtering techniques [11]. Mehdi et al. [12] proposed a hybrid model, Fuzzy Autoregressive Integrated Moving Average (FARIMA), combining ARIMA and fuzzy regression techniques, utilizing the SOFA sliding window strategy to construct a more accurate prediction model for cloud computing traffic forecasting. Anupama [16] proposed a hybrid forecasting model, Seasonal Autoregressive Integrated Moving Average (SARIMA), that integrates statistical and machine learning techniques to predict seasonal and non-seasonal workloads in cloud environments. The linearity of statistical regression models, such as simple and multiple linear regression, is insufficient to capture the time-varying and nonlinear load patterns of cloud resource demands [17–19], and hence such approaches are unsuitable to process such data sequences.

Machine and deep learning and neural networks offer novel approaches to forecast cloud resource demand loads based on time-series data. Unlike statistical regression and signal fitting, machine learning demonstrates greater accuracy for predicting nonlinear cloud resource loads and peaks. Their algorithms have enhanced the precision of load prediction models for cloud resource demand [9,20,21]. The

RCP-CL prediction model uses an integrated parallel and stacked one-dimensional convolutional neural network (1D-CNN) layer adjusted by analyzing the autocorrelation and partial autocorrelation of CPU utilization for kernel size and dilation rate, along with an LSTM network to model random fluctuations for multi-step CPU utilization prediction [21]. A multivariate prediction model using convolutional neural network (CNN) with LSTM forecast CPU, memory, and network usage [22]. Vector autoregression analysis was performed on the input data to filter out linear interdependencies. LSTM modeled irregular time trends in time information. Residual data were computed and fed into a CNN layer to extract the complex features of each virtual machine usage component to achieve accurate multidimensional usage pattern prediction. A load prediction method based on Deep Belief Networks (DBNs) accurately predicts system loads in cloud data centers, to improve profit, satisfy user Service Level Agreements (SLAs) [23], and dynamically allocate resources based on predicted system loads. A Multi-Factor Fuzzy Long Short-Term Memory network (MF-LSTM) combined various mechanisms to construct a cloud proactive autoscaling system [24]. Preprocessing adopted fuzzy techniques to reduce the volatility of monitoring data, while LSTM was utilized to predict multi-variable time-series data for resource consumption. An adaptive window size selection method based on deep learning was proposed to alleviate the issues of inaccurate estimation resulting from significant unrelated observation data for training on a large fixed sliding window, or inaccurate prediction resulting from rapid estimation degradation due to a small window [25], dynamically constraining the sliding window size, capturing the local trend in the latest resource utilization data, and establishing an estimation model for each trend cycle.

With increasingly complex time-series data, researchers have proposed methods to analyze cloud resource load demand data sequences. An online incremental learning method predicted the runtime of tasks in cloud service workflows [26], improving predictive accuracy through task execution sequence features such as CPU utilization, memory usage, and I/O activity. Nawrocki et al. [27] applied machine learning methods and multi-layer perceptron (MLP) models to provide predictive guidance for cloud resource reservation in short- and long-term network services. Osypanka et al. [20] combined machine learning techniques with load prediction, computation service feature extraction, long-term planning for cloud resource utilization, adaptive anomaly detection, and continuous monitoring to establish a cost-effective cloud resource supply plan and generate knowledge for the optimization system and its workload patterns. A data-driven adaptive prediction model for cloud resource usage adaptively adjusted the forecasting pattern to generate short- and long-term cloud resource usage plans [28], accommodating temporary or permanent usage changes with different load characteristics. DTDR-ALSTM improved the performance of an attention-based long short-term memory (ALSTM) network by reconstructing multidimensional data using dynamic time-delay to account for the time-varying impacts of transfer times between industrial process variables during feature extraction [29], utilizing the dynamics between predictive and related variables to enhance the identification of key features extracted from the optimal data. Techniques such as MODALS [30], RGAN, RCGAN, STFT, TimeGAN, GRATIS [31], MAR, MODALS, PBA, and WGAN-gp have been employed to handle time-series data [30,32–36].

Many studies have investigated the correlations among cycles, relative positions, time dependencies, and long-term impact involving a single-variable historical sequence when analyzing time-series data [20,26–28], but neglected to fully capture the complex spatiotemporal dependencies among correlations and the multiple effects of exogenous factors, which generate the sequence data in a latent factor form over different time periods. Ultimately, this limits the understanding of the complex relationships and underlying mechanisms of the generated sequence data. Various approaches have been proposed to predict cloud resource loads based on time-series data, but with limitations such as nonlinearity, multiple unknown latent coupling correlation features, high-dimensional and mutually independent single-variable time-series data, and data sequences with various types of time-series coupling correlations, such as global trends and periodicity. Consequently, the prediction performance of these methods requires further improvement. The prevailing analytical frameworks for cloud resource load forecasting typically incorporate spatiotemporal correlation models based on observable factors in time-series data, but suffer from incomplete information structures, which can hinder the prediction of spatiotemporal dependency features. Current methods cannot effectively integrate spatiotemporal correlation, local feature correlation, and global multidimensional dependency through a single variable when dealing with nonlinear and complex data that may include latent factors and noise. We propose a novel approach to extract potential latent factors by utilizing K-embedding dimension mode decomposition based on a single-variable time series. A multidimensional representation learning framework utilizes a self-attention network to fuse the features of each mode component, to more effectively analyze the dynamics reflected at different temporal granularities within the spatiotemporal structure. By training the proposed framework, the network can learn and explore the inherent spatiotemporal correlation logic rules hidden within the single-variable time-series dataset. The proposed framework encodes different temporal granularities using a multiple-head attention mechanism, effectively extracting and fusing local dependencies and global correlation rules, to fully capture the spatiotemporal dependency characteristics of internal and external factor integration.

Table 1
Notations used in this paper.

Symbol	Description
l_x	The input time series length.
X^t_{en}	The known input of the time series data.
X^t_{out}	The forecast data.
$BIMF^t_i$	The ith modal component by VMD.
SEv	The estimated values of sample entropy for each modal component.
$\phi_k(t)$	A non-monotonically decreasing phase function.
$r_k(t)$	The kth variational mode decomposition.
$\omega_k(t)$	The central frequency of $r_k(t)$.
δ_c	The clustering threshold.
δ_0	The noise threshold.
X^t_{pin}	The Decoder input sequence of the lower branch in the VMDSE-Informer.
S^t_{dev}	The sample entropy-based optimization of the VMD reconstruction subsequence algorithm.

3. VMDSE-Tformer: A cloud resource load prediction model

The time series of a single variable represents the combined effects of multiple related factors. An effective method to explore the relationships between these hidden factors within the time series dataset is to employ multidimensional decomposition and analyze the temporal coupling and correlation between the decomposed variables, as well as their own trend changes. Based on the concept of decomposing the time series data of a certain type of cloud resource load into multiple dimensions, we performed high-dimensional decomposition to explore the inherent multidimensional latent factors. After analyzing the sample entropy (SE) of each component, the time series data were reconstructed based on a Transformer framework to build the cloud resource load prediction model, VMDSE-Tformer.

The important symbols used throughout the paper with their description are listed in Table 1.

Assuming the input time-series length of the prediction model is l_x, which is used as the input length of the encoding module, the known input of the time-series data of a single variable is $X^t_{en} =$

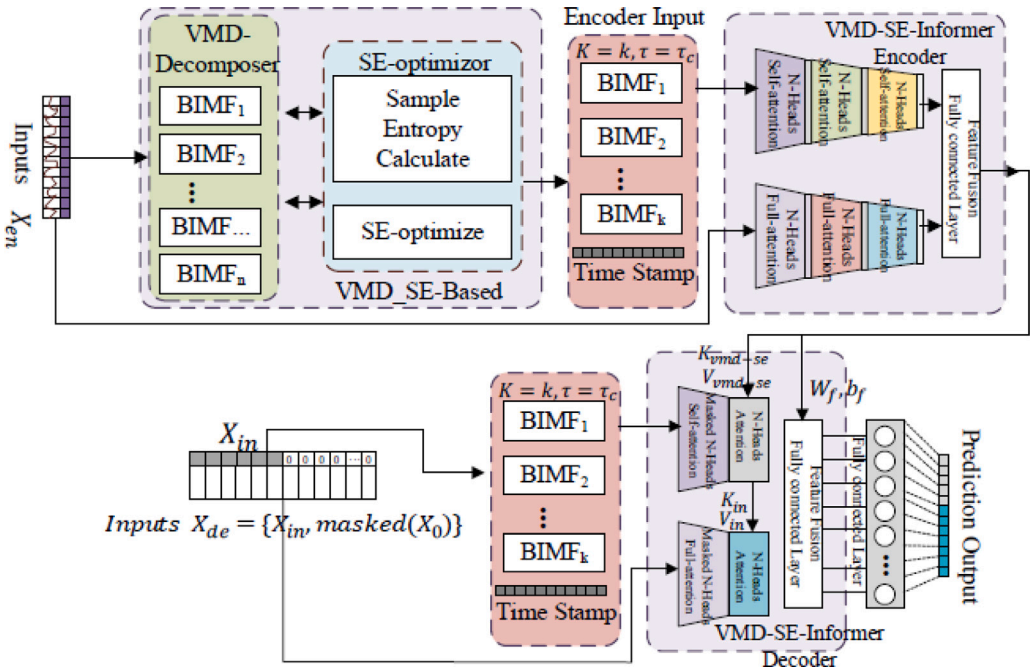


Fig. 1. Overall architecture of the VMDSE-Tformer model.

$\left\{x^t_{t-l_x}, x^t_{t-l_x+1}, \ldots, x^t_{t-1} \mid x^t_i \in \mathbb{R}^1\right\}$, representing the original data of a set of cloud resource demands from time $t - l_x$ to time $t - 1$. We developed the VMDSE-Tformer load prediction model to forecast data $X^t_{out} = \left\{x^t_t, x^t_{t+1}, \ldots, x^t_{t+\lambda-1} \mid x^t_i \in \mathbb{R}^1\right\}$ for the upcoming λ steps based on the historical data of the previous l_x steps' historical data.

3.1. VMDSE-Tformer framework

Fig. 1 shows the framework of VMDSE-Tformer, which has four modules: (1) VMD-Decomposer using VMD to extract latent variables within the time series; (2) SE-optimizer based on subsequence reconstruction using SE to conduct subsequence selection by measuring the self-similarity of each mode component; (3) VMD-SE-Informer Encoder incorporating a Self-Attention mechanism for feature extraction and capture of spatiotemporal interactions across multiple projection spaces of different latent variables; and (4) VMD-SE-Informer Decoder, responsible for subsequence fusion and decoding prediction, capturing temporal coupling associations within multidimensional subsequence data and resulting in reconstructed output predictions. These modules aim to achieve accurate cloud resource load prediction by effectively processing and analyzing the complex spatiotemporal information contained in the time series data.

Fig. 1 shows the training and prediction process of VMDSE-Tformer, along with its overall architecture. VMD-Decomposer uses VMD to decompose the input vector X^t_{en} into n modal components $\{BIMF^t_1, BIMF^t_2, \ldots, BIMF^t_n\}$, $BIMF_n \in r_n(t)$ with different central frequencies, where $BIMF_n \in r_n(t)$ and $r_n(t)$ represents the nth modal component. Due to the limited input length l_x of the prediction model, the SE-optimizer calculates the estimated values of sample entropy (SEv) for each modal component, $SEv = \left\{SEv^t_1, SEv^t_2, \ldots, SEv^t_n\right\}$, whose values are used to optimize the selection of k modal components with overlaid time stamp sequences, and the concatenated subsequence is combined with source data sequence X^t_{en} to generate the reconstructed subsequence to serve as input for VMD-SE-Informer Encoder, which employs two encoders to apply the *Probspare* Self-attention [37] mechanism on $\left\{BIMF_i + TimeStamp\right\}$ and X^t_{en}, which enables the extraction of spatiotemporal coupled correlation features from the subsequence set. Feature fusion is performed on the independently encoded results using a fully connected neural network layer. During the prediction phase, VMD-SE-Informer Decoder receives $X^t_{de} = \left\{X^t_{in}, masked(X_0)\right\}$ as input, where X^t_{in} represents known temporal data with length i_x, and $masked(X_0)$ represents the predicted portion, which consists of a sequence of length λ, with all values set to 0. Using the VMD_SE -Based model, the obtained parameters are utilized to perform the VMD of the X^t_{in} segment, and the resulting sequence is appended with time stamps (Time Stamp) before being combined with X^t_{de} to serve as the input of VMD-SE-Informer Decoder. The decoding process leverages the learned parameters from the VMD-SE-Informer Encoder, which include $\left\{(K_{vmd\_se}, V_{vmd\_se}), (K_{in}, V_{in}), (W_f, b_f)\right\}$, and the prediction is finally generated via a fully connected layer.

To briefly describe the overall data processing process in the model, firstly, the historical sequence is decomposed into n-order modes using the variational mode decomposition method. Then, each mode is analyzed using sample entropy, followed by clustering based on the central frequency of sample entropy for each mode. From the resulting n-order modes, k suitable mode decomposition components are selected and reconstructed into a k-dimensional sequence group, which serves as the input for the prediction model encoding. Consequently, the model takes the reconstructed sequence as a comprehensive input in the form of a k-dimensional sequence group.

3.2. VMD-decomposer

3.2.1. Constructing a variational model

The construction of causal relationship models typically relies on assumptions, and the discovery of the causal structure within dynamic systems from observable time series data carries significant theoretical and practical implications. Cloud resource demand load time series consist of resource requests from various users with different frequencies, and are affected by multiple latent variables, such as requirements, network bandwidths, and time intervals. The sensitivity of hidden causal relationships within time series data to factors like their frequency in the dataset has been demonstrated [38–40]. An effective auxiliary method for mining these is to decompose the cloud resource demand sequence into different frequency scales using modal analysis to enable the identification of frequency characteristics associated with different data acquisition methods. VMD is a fully non-recursive signal processing method that enables adaptive determination of bandwidth

and frequency of signals, effectively decomposing non-stationary multi-component amplitude- and frequency-modulated signals into single-component amplitude and frequency-modulated modal components. VMD addresses the variational problem and enables estimation of corresponding modal components while appropriately balancing errors between modes [41,42]. VMD-Decomposer applies VMD to time series data of cloud resource demand loads to extract modal component information at different frequency scales. Through an iterative search for the best-fit solution, it determines the center frequency and bandwidth of each VMD mode, and adaptively decomposes the data to characterize each hidden variable at its unique frequency.

As shown in Fig. 1, VMD-Decomposer utilizes VMD to decompose the original time-series data of cloud resource demand loads, represented as $f(t) = X_{en}^{t}$, into n modal components. This is achieved by computing n band-limited intrinsic mode functions (BIMFs). The kth VMD is defined as

$$r_k(t) = A_k(t)\cos\left(\phi_k(t)\right) \tag{1}$$

$$\omega_k(t) = \phi_k'(t) = \frac{d\phi_k(t)}{dt}, \tag{2}$$

where $A_k(t)$ is the instantaneous amplitude of the kth modal component $r_k(t)$, $k = 1, 2, \ldots, n$, with central frequency $\phi_k(t)$; $\omega_k(t)$ is a non-monotonically decreasing phase function ; and $t(t > 0)$ represents time.

By applying the Hilbert transform, the one-sided spectrum of the analytic signal of $r_k(t)$ can be obtained as

$$\left[\delta(t) + \frac{i}{\pi t}\right] * r_k(t) \tag{3}$$

where $\delta(t)$ is the unit impulse function and i is the imaginary unit.

Based on Eq. (3), the bandwidth assessment of each modal component can be defined as

$$\left[\left(\delta(t) + \frac{i}{\pi t}\right) * r_k(t)\right] e^{-i\omega_k t}, \tag{4}$$

where ω_k is the central frequency of the kth modal component $r_k(t)$, and $e^{-i\omega_k t}$ is an exponential term indicating the complex central frequency.

Estimates of the bandwidths of modal components $r_k(t)$ are obtained by calculating the $L2$ norm of the gradient of the demodulated signals. The constrained variational model expression constructed for the original input sequence $f(t)$ is

$$\begin{cases} \min_{\{r_k\},\{\omega_k\}} \left\{ \sum_{k=1}^{n} \left\| \partial t \left[\left(\delta(t) + \frac{i}{\pi t}\right) * r_k(t) \right] e^{-i\omega_k t} \right\|_2^2 \right\}. \\ \text{s.t.} \ \sum_{k=1}^{n} r_k(t) = f(t) \end{cases} \tag{5}$$

To obtain the optimal solution of the constrained variational model Eq. (5), it is transformed into an unconstrained variational model problem using quadratic penalty function terms and Lagrange multiplier operators. The resulting extended Lagrangian function is derived as

$$\begin{aligned} L\left(\{r_k\},\{\omega_k\},\lambda\right) = {} & \alpha \sum_{k=1}^{n} \left\| \partial t \left[\left(\delta(t) + \frac{i}{\pi t}\right) * r_k(t) \right] e^{-i\omega_k t} \right\|_2^2 \\ & + \left\| f(t) - \sum_{k=1}^{n} r_k(t) \right\|_2^2 + \left\langle \lambda(t), f(t) - \sum_{k=1}^{n} r_k(t) \right\rangle, \end{aligned} \tag{6}$$

where α is a quadratic penalty factor, $\lambda(t)$ is the Lagrange multiplier operator, and $\langle \bullet \rangle$ represents the inner product.

The optimal solution for the unconstrained variational model in Eq. (6) is obtained by iteratively updating r_k, ω_k, and λ using the alternate direction method of multipliers (ADMM). The corresponding update formulas for each iteration are

$$\hat{r}_k^{m+1}(\omega) = \frac{\hat{f}(\omega) - \sum_{i \neq k} \hat{r}_i(\omega) + \frac{\hat{\lambda}^m(\omega)}{2}}{1 + 2\alpha\left(\omega - \omega_k^m\right)^2} \tag{7}$$

$$\omega_k^{m+1} = \frac{\int_0^\infty \omega \left|\hat{r}_k^{m+1}(\omega)\right|^2 d\omega}{\int_0^\infty \left|\hat{r}_k^{m+1}(\omega)\right|^2 d\omega} \tag{8}$$

$$\hat{\lambda}_k^{m+1}(\omega) = \hat{\lambda}_k^m(\omega) + \tau\left(\hat{f}(\omega) - \sum_{k=1}^{n} \hat{r}_k^{m+1}(\omega)\right), \tag{9}$$

where $\hat{r}_k^{m+1}(\omega)$, $\hat{f}(\omega)$, $\hat{r}_i(\omega)$, and $\hat{\lambda}(\omega)$ are the respective Fourier transforms of $r(t)$, $f(t)$, $r(t)$, and $\lambda(t)$; m is the iteration number, $m+1$ is the $(m+1)$th iterative update, ω_k^{m+1} is the center frequency of the current modal component, and τ is the noise tolerance.

3.2.2. Modal component decomposition algorithm

Based on the fundamental principles of VMD discussed earlier, we present Algorithm 1 $Cal\_u(f(t), \alpha, \varepsilon, \tau)$, which provides an adaptive decomposition of the cloud resource demand load sequence to obtain the kth modal component $r_k(t)$.

Algorithm 1 $Cal\_u(f(t), \alpha, \varepsilon, \tau)$.

Input: $f(t)$, α, ε, τ; The parameters are as follows: cloud resource demand load sequence; the balancing parameter of the data-fidelity constraint; tolerance of the convergence criterion; and noise-slack.
Output: $r_k(t)$,$\omega_k(t)$;
1. Initial $\left\{r_k^1(\omega)\right\}, \left\{\omega_k^1\right\}, \lambda_k^1(\omega), m = 0$;
2. **Do** {
3. $m = m + 1$;
4. Calculate_r $\left(\hat{r}_k^{m+1}(\omega)\right)$; // Eq. (7)
5. Calculate_w $\left(\omega_k^{m+1}\right)$; // Eq. (8)
6. Update_l $\left(\hat{\lambda}_k^{m+1}(\omega)\right)$; // Eq. (9)
7. e $= \frac{\sum_k \left\|\hat{r}_k^{m+1} - \hat{r}_k^m\right\|_2^2}{\left\|\hat{r}_k^m\right\|_2^2}$; // Calculate the tolerance of the convergence criterion.
8. } **While** ($\varepsilon >$ e);
9. **Return** $r_k(t), \omega_k(t)$.

3.3. SE-optimizer

3.3.1. Estimating sample entropy of time series data

Sample entropy (SE) [43] is an improved method to measure the complexity of time-series data. Since SampEn is commonly used to assess the complexity and irregularity of time series in some research, which is often associated with non-stationarity [44], we use it to extract features from the sequences. By computing the SE of each mode component $r_k(t)$ obtained through VMD, we can compare and evaluate their relative complexity for selecting appropriate component subsets for reconstructing the subsequence set from the mode component set.

The similarity tolerance parameter γ ranges from 10% to 25% of the standard deviation of the original data sequence. Assuming a similarity tolerance $\gamma = 0.2$ and an embedding dimension of 2 ($d = 2$), the calculation process of SE can be expressed as Calculate_SE(X_{en}^t, l_x, γ, d):

(1) Given a cloud resource demand data sequence $X_{en}^t = \left\{x_{t-l_x}^t, x_{t-l_x+1}^t, \ldots, x_{t-1}^t \mid x_i^t \in \mathbb{R}^1\right\} = \left\{y_1, y_2, \ldots, y_N\right\}$, where $N = l_x$, we reconstruct it as a matrix:

$$\tilde{Y} = \begin{bmatrix} y_1 & \cdots & y_{N-d+1} \\ \vdots & \ddots & \vdots \\ y_d & \cdots & y_N \end{bmatrix}; \tag{10}$$

(2) The distance between $\tilde{Y}_i$ and $\tilde{Y}_j$ in $\tilde{Y}$ is defined as

$$\begin{aligned} & D_d\left(\tilde{Y}_i, \tilde{Y}_j\right) = \max_{k=0\ldots d-1}\left(\left|\tilde{Y}_{i+k} - \tilde{Y}_{j+k}\right|\right), \\ & 1 \le k \le d-1, 1 \le i \ne j \le N-d+1, \end{aligned} \tag{11}$$

where k is the step size.

(3) For each $\tilde{Y}_i$, we count the number of $j(1 \le j \le N-d, j \ne i)$, which satisfies the condition $D_d\left(\tilde{Y}_i, \tilde{Y}_j\right) \le \gamma$, denoted as

$$B_i^d(\gamma) = \frac{1}{N-d-1} \operatorname{num}\left\{D_d\left(\tilde{Y}_i, \tilde{Y}_j\right) \le \gamma\right\}, 1 \le i \le N-d; \tag{12}$$

(4) We calculate the average value of $B_i^d(\gamma)$ as

$$B^d(\gamma) = \frac{1}{N-d}\sum_{i=1}^{N-m} B_i^d(\gamma); \tag{13}$$

(5) Let $d = d+1$, and repeat steps (1)–(4) to calculate $B_i^{d+1}(\gamma)$ and obtain

$$B^{d+1}(\gamma) = \frac{1}{N-d}\sum_{i=1}^{N-m} B_i^{d+1}(\gamma). \tag{14}$$

Under the similarity constraint parameter γ, the sample entropy is defined as

$$\mathrm{SampEn}(d,\gamma) = \lim_{N\to\infty}\left\{-\ln\left(\frac{B^{d+1}(\gamma)}{B^d(\gamma)}\right)\right\}, \tag{15}$$

where X_{en}^t is a finite time-series of length l_x, so the sample entropy when N is known is

$$\mathrm{SampEn}(d,\gamma,N) = -\ln\left(\frac{B^{d+1}(\gamma)}{B^d(\gamma)}\right). \tag{16}$$

Sample entropy can be used as a measure of non-stationarity [45, 46]. A larger value of $\mathrm{SampEn}(d,\gamma,N)$ indicates higher complexity of the sequence.

3.3.2. Subsequence reconstruction using sample entropy-based strategy

Based on n modes components $\{BIMF_1, BIMF_2, \ldots, BIMF_n \mid BIMF_n \in r_n(t)\}$ obtained from VMD, we derived the corresponding SampEn (d,γ,l_x) values $SEv = \{SEv_1^t, SEv_2^t, \ldots, SEv_n^t\}$. By referencing SEv_0^t of the original data sequence X_{en}^t, a subset of subsequences was selected. We chose SEv_i^t that were both similar and far from the corresponding ith mode component in SEv_0^t. We excluded subsequence subsets with values close to or greater than that of SEv_0^t, and utilized this subset, together with the original sequence, as input to the encoder to extract features for reconstructing the original sequence. Thus, we propose a sample entropy-based subsequence reconstruction method to obtain causal relationship features of the latent variables in the original data sequence by optimally solving for two core parameters.

(1) To select the appropriate number of VMD mode components, the central frequency $\omega_n(t)$ obtained by performing VMD on the original data sequence changes with the number of mode components n. As the decomposed mode components approach a specific value, $\omega_k(t)$ tends to converge. The presence of mode mixing is more likely to occur when two adjacent mode components possess similar central frequencies. Empirical observations suggest that increasing the number of VMD mode components n can lead to less distinct features of the original data's hidden variables. To avoid the loss or mixing of modes while obtaining better latent variable features, we start the search with an initial number of mode components $n_0 = 3$.

Assuming $k(k \geq 3, k \in N)$ mode components, the maximum central frequency among these k components is $\omega_k^{\max}(t) = \max\left\{\omega_i^k(t)\right\}, (i = 1,2\ldots,k)$, while the minimum central frequency is $\omega_k^{\min}(t) = \min\left\{\omega_i^k(t)\right\}, (i = 1,2\ldots,k)$. The optimal number of VMD mode components for this given raw data is $n = k$ if k simultaneously satisfies two conditions:

$$\begin{aligned}&\left(\left(\omega_k^{\max}(t) - \omega_{k-1}^{\max}(t)\right) \gg \left(\omega_{k+1}^{\max}(t) - \omega_k^{\max}(t)\right)\right)\\ &\text{and } \left(\left(\omega_{k+1}^{\max}(t) - \omega_k^{\max}(t)\right) \approx \left(\omega_{k+2}^{\max}(t) - \omega_{k+1}^{\max}(t)\right)\right)\end{aligned} \tag{17}$$

$$\begin{aligned}&\left(\left(\omega_{k-1}^{\min}(t) - \omega_k^{\min}(t)\right) \gg \left(\omega_k^{\min}(t) - \omega_{k+1}^{\min}(t)\right)\right)\\ &\text{and } \left(\left(\omega_k^{\min}(t) - \omega_{k+1}^{\min}(t)\right) \approx \left(\omega_{k+1}^{\min}(t) - \omega_{k+2}^{\min}(t)\right)\right).\end{aligned} \tag{18}$$

(2) We choose an appropriate subset ϕ from the known VMD mode components by determining the optimal number of mode components n using step (1), and then, according to the SampEn (d,γ,l_x) values that correspond to the n mode components, and based on the following conditions, select a suitable subset of $\phi = \left\{r_\mu^n(t) \mid \mu \in \{1,2,\ldots,n\}\right\}$ from the n mode components.

As described in Section 3.3.1, let $SEv = \{SEv_1^t, SEv_2^t, \ldots, SEv_n^t\}$ represent the SampEn (d,γ,l_x) values corresponding to the n mode components. We denote the $SampEn$ of the original data sequence as SEv_0^t. We select the reconstructed subsequence set based on the following conditions, with δ_c and δ_o serving as the respective clustering and noise thresholds:

- When $\left|SEv_0^t - SEv_i^t\right| < \delta_c$, then $r_i^n(t) \notin \phi$. To avoid overfitting during training, and since the reconstructed subsequence set contains the original data sequence, the mode components with $SampEn$ values less than δ_c compared to the original data sequence can be neglected;
- When $SEv_j^t - SEv_0^t \gg 2\delta_o$, then $r_j^n(t) \notin \phi$. Mode components with $SampEn$ values far from and larger than $2\delta_o$ compared to SEv_0^t are considered to be noise, and can also be discarded;
- $\phi = \left\{r_\mu^n(t) \mid \mu \in \{1,2,\ldots,n\}\right.$. and $\mu \neq i$ and $\left.\mu \neq j\right\}$.

The pseudocode for Algorithm 2, $OVS\_Seqset(d,\gamma,f(t))$, is based on the VMD SampEn-optimized reconstructed subsequence algorithm.

Algorithm 2 $OVS\_Seqset(d,\gamma,f(t))$

Input: $d,\gamma,f(t)$;//The parameters are as follows: the embedding dimension $d = 2$; the similarity tolerance $\gamma = 0.2$; the time series $f(t)$.

Output: ϕ; //The subset ϕ from the n mode components.

1. Initial $k = 3, d = 2, \gamma = 0.2, Conditions = False$;
2. **For** $(i = 1 to k+2)$ $\quad Cal\_u(f(t),\alpha,\epsilon,\tau)$;
3. **While** (Conditions=$False$) {
4. $\quad \omega_k^{\max}(t) = \mathrm{Getmax}\,\omega()$;
5. // $\quad \omega_k^{\max}(t) = \max\left\{\omega_i^k(t)\right\}, (i = 1,2\ldots,k)$.
6. $\quad \omega_k^{\min}(t) = \mathrm{Getmin}\,\omega()$;
7. // $\quad \omega_k^{\min}(t) = \min\left\{\omega_i^k(t)\right\}, (i = 1,2\ldots,k)$
8. $\quad$ **If** Conditions $satisfied$ // Eq. (17) - Eq. (18) are satisfied.
9. $\quad\quad$ Conditions=Ture;
10. $\quad\quad$ $n = k$; // Obtaining the optimal number of VMD modal components n.
11. $\quad$ **Else** $k = k+1$;
12. $\quad\quad$ $Cal\_u(f(t),\alpha,\epsilon,\tau)$; // Get the VMD set of $k+1$.
13. } // End of $While(Conditions = False)$.
14. **For** $(i = 1 to n)$ $\quad SampEn(d,\gamma,l_x,i)$;
15. $SampEn(d,\gamma,l_x,0)$;
16. $\delta = Cal\_Threshold(SEv[])$: // Calculating the δ_c,δ_o;
17. If $\left(\left|SEv_0^t - SEv_i^t\right| < \delta_c$ or $SEv_j^t - SEv_0^t \gg 2\delta_o\right)$;
18. $\phi\backslash i,j$; // Discarding the corresponding modal components.
19. **Return** ϕ.

Based on the VMD sample entropy, the reconstruction subsequence algorithm is optimized to obtain the VMD modal component subset, which is merged with the original data sequence to form the input of the predictive model for reconstruction subsequence $S(S \in \mathbb{R}^{(|\phi|+1)\times l_x}, k = |\phi|)$, as illustrated in Fig. 1.

3.4. VMD-SE-informer

The reconstructed subsequences, optimized based on the VMD sample entropy, are encoded using the multi-layer multi-head self-attention module of Transformer, designed to capture the interactions and features of hidden factors in a single time-series across different temporal and spatial dimensions in multiple projection spaces. As shown in Fig. 1, the input of VMD-SE-Informer is $S = \left\{BIMF_1, BIMF_2, \ldots, BIMF_k, X_{en}^t\right\}, k = |\phi|$, where $BIMF_i \in \mathbb{R}^{1\times l_x}, S \in \mathbb{R}^{(k+1)\times l_x}, i = 1,2,\ldots,k$.

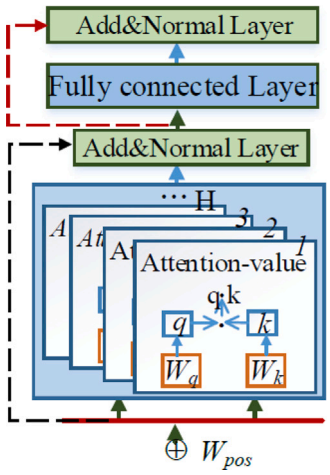


Fig. 2. Transformer encoder.

3.4.1. VMD-SE-informer encoder

Two branches of three-layer multi-head attention modules are constructed within the VMD-SE-Informer Encoder, each layer comprising a multi-head self-attention encoder. The initial input sequence of encoder $S^{(0)}$ is augmented with position encoding W_{pos}, such that each element s_t is represented as $s_t + W_{pos_t}$. So, we define

$$S^{(0)} = \left\{ s_1 + W_{\text{pos }_1}, s_2 + W_{\text{pos }_2}, \ldots, s_T + W_{\text{pos }_T} \right\}, \tag{19}$$

where $W_{\text{pos }_t} \in \mathbb{R}^D, D = |\phi|$ is the number of sequences in the reconstruction subsequence set.

As shown in Fig. 2, during the encoding process of the multi-head self-attention encoder in each layer, multiple trainable matrices W_q^h, W_k^h, W_v^h, h are applied to the self-attention model in H projection spaces. The *Query*, *Keys*, and *Values* in the corresponding projection spaces are calculated as

$$\begin{aligned} &\text{MultiHead }(S) = W_0 \left[\text{ head }_1; \ldots, \text{ head }_H \right] \\ &\text{head}_h = \mathcal{A}\left(Q_h, K_h, V_h\right) = \text{softmax}\left(\frac{Q_h K_h^T}{\sqrt{d_k}}\right) V_h \\ &\begin{cases} Q_h = W_q^{(h)} S \\ K_h = W_k^{(h)} S \\ V_h = W_v^{(h)} S \end{cases} \quad \forall h \in \{1, 2, \ldots, H\}, \end{aligned} \tag{20}$$

where d_k is the dimension of the column vectors of input matrices Q_h and K_h, d_v is the dimension of column vectors of V_h, $W_0 \in \mathbb{R}^{D \times H d_v}$ is the output projection matrix, and $W_q^{(h)} \in \mathbb{R}^{D \times d_k}$, $W_k^{(h)} \in \mathbb{R}^{D \times d_k}$, $W_v^{(h)} \in \mathbb{R}^{D \times d_v}$ are trainable matrices.

As shown in Fig. 1, the VMD-SE-Informer Encoder comprises three Transformer encoders of different scales for the original data sequence and reconstruction subsequence set. Each encoding layer is calculated using a multi-head self-attention module and a nonlinear $FNN(\cdot)$ feedforward neural network applied on a per-position basis. In Fig. 2, residual connections are applied to connect the output of the previous layer to the current layer during the computation of each layer in the encoder, followed by layer normalization. $FNN(\cdot)$ refers to a fully connected layer with two layers, connected by a *ReLu* activation function, which is defined as

$$FNN(v) = W_{2FNN}\,\text{ReLU}\left(W_{1FNN} v + b_{1FNN}\right) + b_{2FNN}, \tag{21}$$

where $v \in S^{(l)}$ denotes the vector at each position in the input sequence of the previous layer, W_{1FNN} and W_{2FNN} are the weight matrices of the two-layer neural network, and b_{1FNN} and b_{2FNN} are the biases of the two corresponding layers, which are all trainable network parameters. The connection weights between layers in the encoder are calculated dynamically by the self-attention mechanism.

3.4.2. The internal structure

As traditional Transformer training utilizes a multi-head self-attention mechanism [47], it consumes a large amount of training time and memory. To improve the training efficiency, reduce the time, and minimize the consumption of server memory while maintaining training and prediction performance, VMD-SE-Informer adds 1D convolutional layer compression between every two layers of the three encoders. A dual self-attention mechanism is adopted in each encoder, and sampling-style attention calculation is utilized on the reconstruction subsequence set to reduce the computational complexity during training.

(1) Dual Self-attention Mechanism Calculation Model for Internal Self-attention Strategy

We adopt a calculation model with a dual self-attention mechanism, which improves training efficiency without reducing model accuracy [37]. For the encoder that processes the reconstruction subsequence set S, the attention estimation formula for the ith query in each self-attention is

$$E\left(q_i, K\right) = \max_j \left\{ \frac{q_i k_j^T}{\sqrt{d_k}} \right\} - \frac{1}{l_x} \sum_{j=1}^{l_x} \frac{q_i k_j^T}{\sqrt{d_k}}. \tag{22}$$

We compute the attention using $E\left(q_i, K\right)$ for each position of the sequence. Then, we retrain the μ highest $E\left(q_i, K\right)$ values obtained from the sampled positions, where $\mu = \alpha \ln l_x$, and $\alpha = 5$. The rest $\left(l_x - \mu\right)$ positions is averaged as

$$\bar{E}\left(q_i, K\right) = \frac{1}{l_x} \sum_{j=1}^{l_x} \frac{q_i k_j^T}{\sqrt{d_k}} = \text{mean}(V). \tag{23}$$

The self-attention formula for the encoder that processes the reconstruction subsequence set S can be derived from Eqs. (22) and (23) as

$$\mathcal{A}_S\left(Q_h, K_h, V_h\right) = \text{softmax}\left(\frac{\overline{Q_h} K_h^T}{\sqrt{d_k}}\right) V_h. \tag{24}$$

$\overline{Q_h}$ has two components calculated using Eqs. (23) and (24). The first component corresponds to the top $\mu E\left(q_i, K\right)$ values at their corresponding positions, while the values corresponding to the remaining positions are averaged to obtain the second component, $\bar{E}\left(q_i, K\right)$. This attention calculation method [37,48] is commonly used to provide a good balance between accuracy and efficiency.

To exploit the inherent correlation of the original sequence X_{en}^t as much as possible, the full self-attention calculation formula is preserved for the self-attention of the X_{en}^t processing encoder,

$$\mathcal{A}_X\left(Q_h, K_h, V_h\right) = \text{softmax}\left(\frac{Q_h K_h^T}{\sqrt{d_k}}\right) V_h. \tag{25}$$

(2) The Internal Structure of Encoder

The VMD-SE-Informer Encoder handles the reconstruction subsequence S and original univariate sequence X_{en}^t in separate branches. We apply different self-attention strategies to capture the intrinsic spatiotemporal features in these two sequences, *i.e.*, the downsampled $Top - \mu$ self-attention strategy (Eq. (24)) for the reconstruction subsequence, and the Full-attention strategy (Eq. (25)) for the original univariate sequence. The branches and feature compression structure of the VMD-SE-Informer Encoder are shown in Fig. 3.

As illustrated in Fig. 3, we employ a novel approach to extract feature information with varying emphasis from different sequences and improve training efficiency while reducing the memory consumption of the server [37]. By leveraging the compression capability of convolutional networks [49], we incorporate a one-dimensional convolution ($Conv1D$) in two branches. We employ Max Pooling and Average Pooling in two branches as shown in Fig. 3. The Max Pooling in the upper branch extracts feature texture, specifically emphasizing the extraction of spatiotemporal edge feature information that contains the

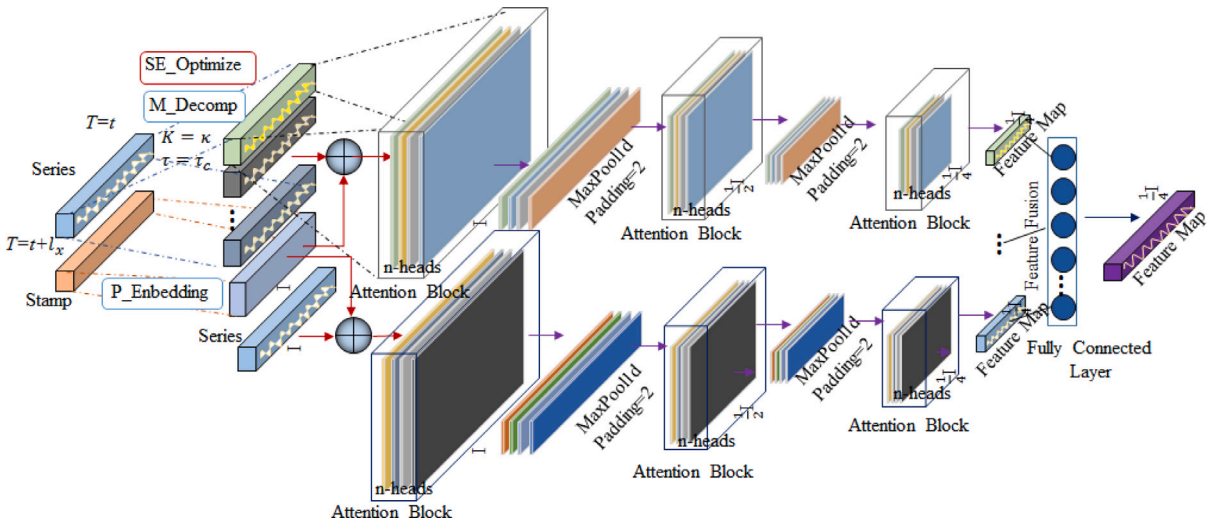


Fig. 3. The internal structure of VMD-SE-Informer.

demand mutation in the reconstructed subsequence S. In the lower branch, the Average Pooling preserves inherent smoothness and feature information related to resource demand in the sequence, with a focus on retaining the overall data characteristics of the original univariate sequence X_{en}^{t} and transmitting it to the next sampling layer.

According to the structure of the Encoder in Fig. 3, data compression and information transfer between layers l and $(l+1)$ in the two branches are defined as

$$S^{(l+1)} = \mathrm{MaxPool1d}\left(\mathrm{ELU}\left(\mathrm{Conv}\ 1d\left(\left[S^{(l)}\right]_{ABlock}\right)\right)\right) \tag{26}$$

$$X_{en}^{t\ (l+1)} = \mathrm{AvgPool1d}\left(ELU\left(\mathrm{Conv}\ 1d\left(\left[X_{en}^{t\ (l)}\right]_{ABlock}\right)\right)\right), \tag{27}$$

where MaxPool1d(·) and AvgPool $1d(\cdot)$ respectively denote Max Pooling and Average Pooling, with $kernel\_size = 3, stride = 2, padding = 1$. The activation function $ELU(\cdot)$ is

$$F(x) = \begin{cases} x, & x > 0 \\ \alpha\left(e^{x} - 1\right), x \le 0 \end{cases}. \tag{28}$$

Conv1d(·) represents a one-dimensional CNN, and $[\cdot]_{ABlock}$ refers to the Multi-Head Attention module in the corresponding encoding layer.

These two branches extract different data features and fuse them through a fusion layer, and then output by a fully connected layer.

$$M_{\mathrm{out}} = W_{\mathrm{out}}\left(\mathrm{Concat}\left(\bar{S}, \overline{X_{en}^{t}}\right)\right) + b_{\mathrm{out}}, \tag{29}$$

where $M_{\mathrm{out}}, \bar{S}, \overline{X_{\mathrm{en}}^{t}} \in \mathbb{R}^{\frac{1}{4}l_x}$, where W_{out} is the weight matrix of the fusion layer, b_{out} is the bias matrix of the fusion layer, $\bar{S}$ and $\overline{X_{en}^{t}}$ are the respective feature outputs of the upper and lower branches of the Encoder, as shown in Fig. 3, and Concat(·) represents sequence concatenation.

3.4.3. Decoder and training optimizer

In the prediction model, the length of the known input sequence used for guidance is l_{pi}, and the length of the predicted output sequence is l_{po}. X_{pin}^{t} is the Decoder input sequence of the lower branch in VMD-SE-Informer, with length $l_{pi} + l_{po}$. The first l_{pi} elements are known as X_{pi}^{t}, and the last l_{po} elements are initialized as 0, and referred to as X_{po}^{t}. Hence we can write

$$X_{pin}^{t} = \mathrm{Concat}\left(X_{pi}^{t}, X_{po}^{t}\right) \in \mathbb{R}^{1\times(l_{pi}+l_{po})}. \tag{30}$$

According to the processing procedure in Fig. 2, we employ the sample entropy-based optimization of the VMD reconstruction subsequence algorithm, Algorithm 2 $OVS_Seqset(\cdot)$, to process X_{pi}^{t} as

$$S_{\mathrm{dev}}^{t} = OVS\_\mathrm{Seqset}\left(d, \gamma, X_{pi}^{t}\right) \in \mathbb{R}^{(|\phi|+1)\times l_{pi}}, \tag{31}$$

where ϕ is the reconstructed subsequence set obtained by Algorithm 2 $OVS\_Seqset(\cdot)$; S_{dev}^{t} is the input reconstructed subsequence of the Decoder in Fig. 2 after formatting, and $(|\phi|+1)$ is the dimension of the reconstructed subsequence.

The objective of the prediction model is to forecast the resource demand for a future period of length l_{po}. To achieve this, we obtain the reconstructed subsequence set S_{dev}^{t} from the historical data sequence of the known guidance portion through VMD sample entropy optimization. The input sequence S_{dein}^{t} of the upper branch's Decoder is formed by concatenating the Mask sequence S_{de0}^{t}, with length l_{po}, with each data element initialized to 0. Define

$$S_{\mathrm{dein}}^{t} = \mathrm{Concat}\left(S_{\mathrm{dev}}^{t}, S_{\mathrm{de}\ 0}^{t}\right) \in \mathbb{R}^{(|\phi|+1)\times(l_{pi}+l_{po})}. \tag{32}$$

The input sequences for prediction are generated by adding a time dimension parameterized by date and time to X_{pin}^{t} and S_{dein}^{t}, respectively.

Let $\hat{y}$ denote the model's predicted output sequence, and y the original data sequence. According to the prediction objective of VMD-SE-Informer, we aim to forecast the resource scheduling demand for the future τ periods. The loss function of the prediction model is

$$\mathcal{L}_{OSS} = \frac{1}{n}\sum \|\hat{y} - y\|^{2} \quad \hat{y}, y \in \mathbb{R}^{\tau}. \tag{33}$$

To accelerate convergence during training, we apply the backpropagation (BP) algorithm on the partial derivative of the loss function value, denoted as $\Delta\delta^{(r)}$, of the prediction model at the rth round of training to learn the network parameters. Furthermore, we optimize the model's parameters using the Adam optimizer,

$$\rho^{(r+1)} = \rho^{(0)}\left(\frac{1-\beta_2^{r}}{1-\beta_1^{r}}\right)^{1/2} \tag{34}$$

$$\theta^{(r+1)} = \beta_1\theta^{(r)} + \left(1-\beta_1\right)\Delta\delta^{(r)} \tag{35}$$

$$\vartheta^{(r+1)} = \beta_2\vartheta^{(r)} + \left(1-\beta_2\right)\left(\Delta\delta^{(r)}\right)^{2} \tag{36}$$

$$\delta^{(r+1)} = \delta^{(r)} - \rho^{(r+1)} * \theta^{(r+1)} / \left(\sqrt{\vartheta^{(r+1)}} + \varepsilon\right), \tag{37}$$

where $\rho^{(0)}$ is the initial learning rate, which is 0.01 during training. The learning rate $\rho^{(r+1)}$ for the next round is calculated based on the initial learning rate $\rho^{(0)}$ using Eq. (34). $\theta^{(r+1)}$ and $\vartheta^{(r+1)}$ are respectively the first-order exponential smoothing values of the gradients and the first-order smoothing values of the gradients squared during the training process. $\delta^{(r+1)}$ is the updated network parameter set acquired by the prediction model in the new round, which is calculated using Eq. (37). The initial values of $Adam$ optimizer parameters β_1, β_2, and ε for the prediction model are set to 0.9, 0.999, and 10^{-8}, respectively.

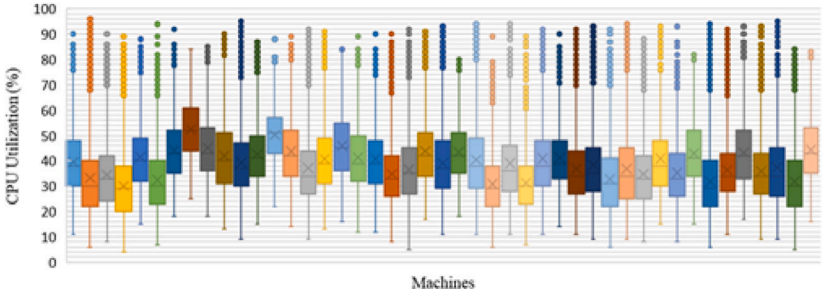


Fig. 4. Box plots of CPU utilization for 45 randomly selected servers in Alibaba Cluster Trace dataset.

4. Data reconstruction and formatting

4.1. Dataset and prediction objective

To forecast user demand for cloud resources, we use the two most commonly used datasets in cloud resource scheduling research, *i.e.*, Google Cluster Trace [50] and Alibaba Cluster Trace,[1] as test experimental datasets to validate the proposed algorithm and the accuracy of the prediction model. Google Cluster Trace contains the workload data of about $25,462,157$ scheduling tasks executed by more than 670,000 workloads on approximately 12,000 physical servers during a 29-day period. Alibaba Cluster Trace contains the task scheduling data for 4023 servers during an 8-day period. For Alibaba Cluster Trace, we randomly selected 45 servers and observed and recorded their CPU utilization during each time period. The box plot results of the statistical analysis are presented in Fig. 4.

According to the statistics, the majority of the CPUs in the servers have utilization rates ranging from 20% to 50%, with an average of approximately 27%. Therefore, it is more effective to allocate cloud resources to predict the workload intensity, CPU, and memory needs of the entire data center, so as to better utilize idle resources such as CPU and memory. By applying the cloud resource scheduling prediction model and extracting the historical data series features in different time windows of Google Cluster Trace, we predicted the corresponding workload intensity, CPU, and memory utilization rates for a future time period.

Based on Google Cluster Trace, if we set the length of the time window to 2 min and 5 min, we can obtain data sequences with respective lengths of 20,880 and 8352 data points, showing the workload intensity, CPU, and memory utilization rates for the entire data center in the corresponding time window. Depending on the resource scheduling window requirements of the data center, the input length of the prediction model can be set as 60, and the output length can be set as 60, 30, or 15. This corresponds to the predicted workload intensity, CPU, and memory utilization rates for the upcoming 1 h and 2.5 h, respectively, based on historical data with input time lengths of 2 and 5 h.

4.2. Parameter selection and optimization for VMD

The number of VMD mode components has a significant impact on the reconstructed subsequence. In the prediction model, the input is only a small segment of the historical data in the source data sequence. Here, a random interval sampling method is applied to extract successive data sequences with a length of P_e from the source data sequence of length L_s as a sample. The reasonable selection rule for the number of VMD mode components proposed in Section 3.3.2 is used to analyze the optimal selection parameter of the number of VMD mode components.

[1] https://github.com/alibaba/clusterdata/blob/master/cluster-trace-v2018/trace_2018.md.

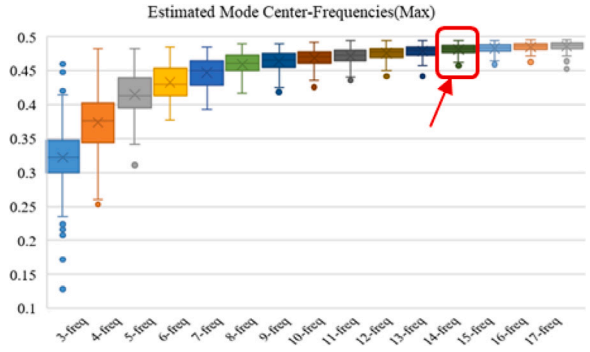


Fig. 5. Box plots of the distribution of the highest central frequencies of various orders of VMD mode components obtained by random interval sampling.

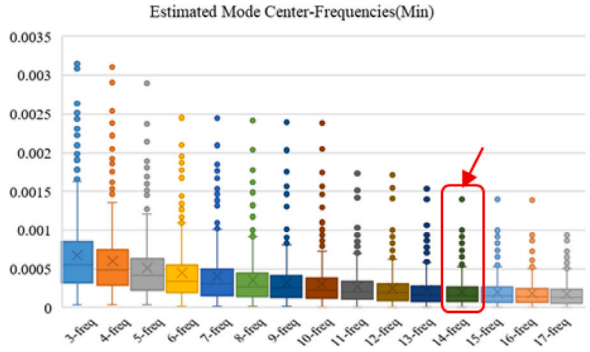


Fig. 6. Box plots of the distribution of the lowest central frequencies of various orders of VMD mode components obtained by random interval sampling.

To ensure random and independent samples, the sampling interval in the sequence should not be set too small; otherwise, there may be large correlations among the samples, which could affect the reliability of the analysis. Random interval sampling can generate samples with different intervals, and the number of samples is $m = 3 * {}^{L}{}_{s}/P_e$, ensuring that the mean follows a normal distribution and is representative. Here, L_s is the length of the source data sequence, and the length of a sample sequence is P_e. When analyzing the task sequence of Google Cluster Trace, $P_e = 120$ is taken as the sample in a length of $L_s = 14616$ (70% of the source data sequence is used as the training set), and the VMD mode component number n is selected for analysis.

There were 363 samples obtained by random interval sampling, and Algorithm 1 $Cal\_u(f(t), \alpha, \epsilon, \tau)$ was applied to obtain the highest and lowest central frequencies of each order of VMD mode components, with $n \in \{3, 4, \ldots, 17\}$. The distribution box plots of the highest and lowest central frequencies are shown in Figs. 5–6.

After obtaining the central frequencies of each order of VMD mode components, the difference in central frequency between adjacent two-level mode components was further calculated, and the resulting plot

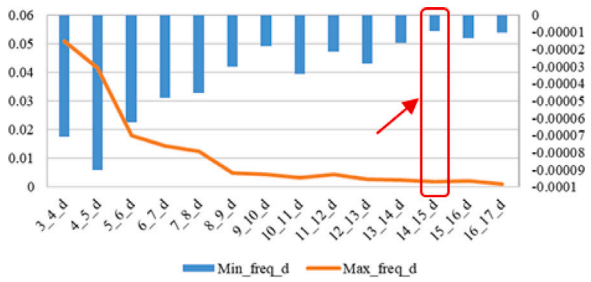


Fig. 7. Line chart of difference in central frequency between adjacent VMD modal components.

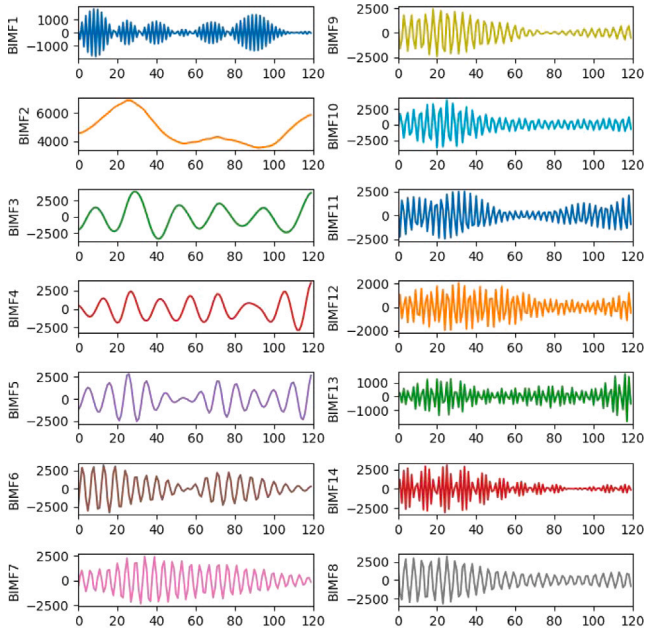


Fig. 8. Line charts of all 14-order VMD modal components for the randomly selected sample.

is shown in Fig. 7, where the curve is the line chart of the difference in the highest central frequency between adjacent VMD modal components, and the histogram represents the difference in the lowest central frequency between adjacent VMD modal components. According to the rules of selecting the number of VMD modal components, and based on the statistical results of Figs. 5–7, it can be known that when the corresponding number of VMD modal components is $n = 14$, the distributions of both the highest and lowest central frequencies tend to be stable, and the subsequent increase in the number of modal components has little influence on the central frequency. As shown in Fig. 7, the absolute value of the central frequency difference between $n = 14$ and $n = 15$ (denoted as "14_15_d", representing the central frequency difference between modal components BIMF14 and BIMF15) tends to be closer to 0; hence to ensure that VMD has stronger feature distinctiveness and more meaningful analysis for predicting the task sequence intensity of Google Cluster Trace, the number of VMD modal components was selected as $n = 14$.

An adjacent data sequence of length 120 was taken from the task sequence of Google Cluster Trace (with a time interval window of 2 min) and decomposed into various-order modal component figures based on VMD, with a modal component number of 14, as shown in Fig. 8. While some BIMFs shown in Fig. 8 may exhibit unconventional patterns, their decomposition is based on the optimization of a variational principle, which may not prioritize intuitive interpretability.

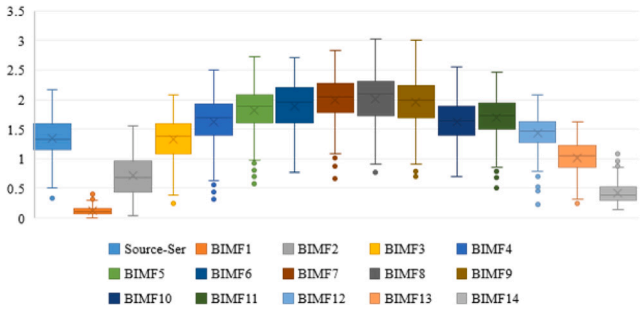


Fig. 9. Boxplot of sample entropy distribution of VMD modal components.

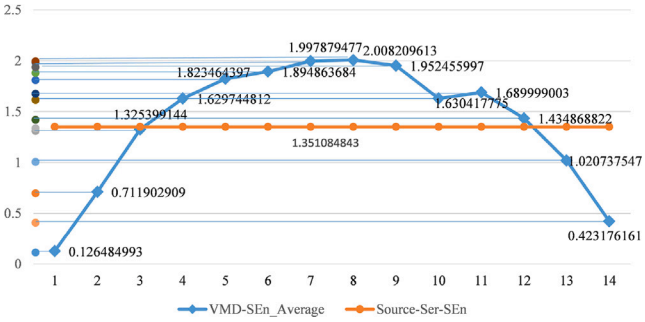


Fig. 10. Line chart of sample entropy means and clustering diagram of sample entropy values.

When analyzing and predicting the CPU and memory scheduling of Google Cluster Trace, the same rule was applied to select the number of VMD modal components for a time window of 2 min, with the numbers being 7 and 9, respectively.

4.3. Reconstruction of subsequences

VMD is applied to decompose the dataset sequences into modal components so as to fit the important characteristic factors inherent in the original data sequences as closely as possible across different spatial and frequency domains. After obtaining the VMD modal components of the cloud resource demand sequence, each contains different amounts of valid information, and a given one may also be a noise component. Therefore, it is necessary to select the modal component with the maximum amount of valid information as much as possible, so that the constructed modal component set can effectively restore the characteristics of the original data sequence and improve the effectiveness of the reconstructed subsequences for the prediction model. Here, the sample entropy of each order modal component is calculated, and the subsequences are reconstructed after clustering based on the sample entropy results using Algorithm 2 $OVS\_Seqset(d, \gamma, f(t))$.

The task sequences of Google Cluster Trace were sampled using the method described in Section 4.2, resulting in 363 samples. The sample entropy of the 14-order VMD modal components corresponding to each sample was calculated. Fig. 9 shows the boxplots of the sample entropy distribution for the original data sequence and all sample sequences. Fig. 10 shows the corresponding line chart of the sample entropy mean values and the clustering diagram of their sample entropy values. The sample entropy means of the VMD modal components for each order corresponding to all samples were clustered as shown in Table 2.

Fig. 10 presents the differences between the sample entropies of each order by plotting the sample entropy (line) of the 14-order VMD modal components and of the original data sequence (straight line), and the scatter plot on the left side of Fig. 10 is obtained by mapping each sample entropy to the vertical axis. The clustering results among

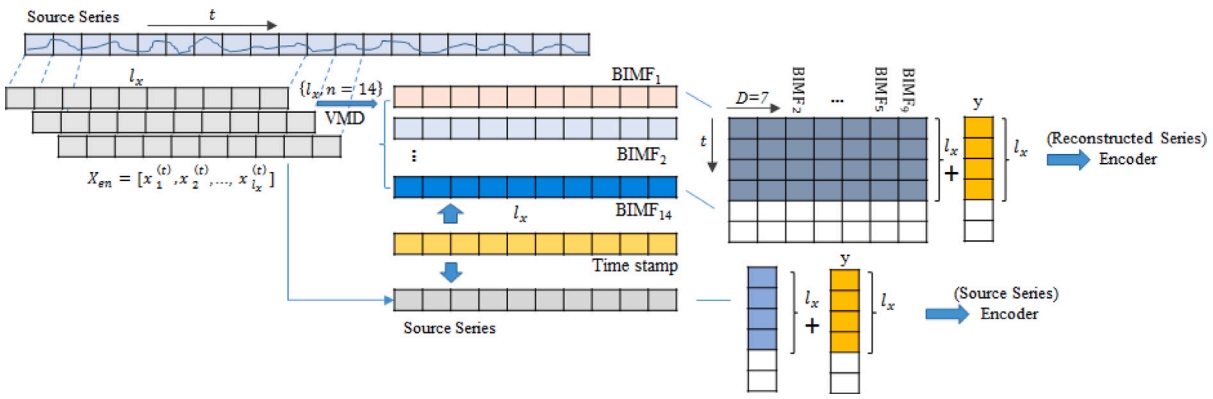


Fig. 11. Formatting of encoder input data.

Table 2
Summary and clustering table of sample entropy mean values of VMD modal components for each order.

Class	Model series	VMD-SEn_Avg	Kernel
1	BIMF1	0.1265	BIMF1
2	BIMF14	0.4232	BIMF14
3	BIMF2	0.7120	BIMF2
4	BIMF13	1.0207	BIMF13
5	BIMF3 Source-series BIMF12	1.3254 1.3511 1.4349	Source-series
6	BIMF4 BIMF10 BIMF11	1.6297 1.6304 1.6899	BIMF10
7	BIMF5	1.8235	BIMF5
8	BIMF6 BIMF9 BIMF7 BIMF8	1.8949 1.9525 1.9979 2.0082	BIMF9

sample entropies are observed. According to the subsequence reconstruction strategy based on sample entropy proposed in Section 3.3.2, the clustering threshold δ_c and noise threshold δ_o were calculated at the 95% confidence level based on the sample entropy mean $SEn_{ss} = 1.351$ of the original data sequence (source-series), which gives $\delta_c = (SEn_{sS} * 95\%)/n = 1.351 * 95\%/14 = 0.092$ and $\delta_o = SEn_{ss} * 95\% = 1.351 * 95\% = 1.284$, where n is the number of VMD modal components, taken as 14. By clustering with the minimum number of classes, the 14-order VMD modal components are divided into the minimum of 8 classes using $\delta_c = 0.092$, as shown in Table 2.

As shown in Table 2, since $\delta_o = 1.284$, the sample entropies of BIMF1–BIMF14 are not far from the sample entropy of the original data sequence; hence all categories are valid. In addition, since the sample entropies of $BIMF3$, $Source-series$, and $BIMF12$ are close, *i.e.*, they have similar features compared to the original data sequence, sequences in the same category as the original sequence can be discarded. Finally, the reconstructed subsequence set contains seven subsequences, *i.e.*, $K = 7$, in the encoder input of Fig. 1. The corresponding categories of data sequences are {BIMF1(BIMF1)}, {BIMF14(BIMF14)}, {BIMF2(BIMF2)}, {BIMF13(BIMF13)}, {BIMF4, BIMF10, BIMF11 (BIMF10)}, {BIMF5(BIMF5)}, and {BIMF6, BIMF9, BIMF7, BIMF8 (BIMF9)}, where parentheses indicate the clustering nucleus. Cloud resource demand changes are complex, and are affected by multiple factors, exhibiting nonlinearity, non-stationarity, and complexity. To make the features of the reconstructed subsequences more prominent, clustering nucleus sequences are selected as class representatives within the same category, and sequences outside the clustering nucleus within the same category can be discarded. As a result, the reconstructed subsequence set of task sequences in Google Cluster Trace is Series $Set = \{BIMF1, BIMF14, BIMF2, BIMF13, BIMF10, BIMF5, BIMF9\}$. A similar analysis can be applied to the CPU and memory demand sequences for Google Cluster Trace, as described above.

4.4. Data formatting

When training the prediction model for cloud resource scheduling, the input sequence X_{en} of the Encoder has length l_x. As $K = 7$ is obtained from the VMD modal decomposition in Section 4.3, the Encoder has two branches, as shown in Figs. 1 and 3, for the original data sequence and reconstructed subsequence set, respectively. The processing and transformation of the data format are illustrated in Fig. 11, where the original task sequence is univariate time-series data that undergo $VMD(l_x, n = 14)$ modal decomposition with a specified K value ($K = 7$) in Section 4.3 to obtain the 14-order modal components. K modal components are selected and transformed with timestamps to obtain the $\mathbb{R}^{l_x \times K}$ representation, which serves as the input for the Reconstructed Series Encoder. The other part of the input is the original data sequence with length l_x, which is directly transformed with timestamps to obtain the $\mathbb{R}^{l_x \times 1}$ representation and serves as the input for the Source Series Encoder.

5. Experiments and results analysis

5.1. Experimental objectives and evaluation metrics

Two experiments were conducted to verify the effectiveness of VMDSE-Tformer, the proposed prediction model for cloud resource scheduling based on a sample entropy modal decomposition attention mechanism. (1) The effectiveness and validity of the model structure were verified through ablation experiments, where the prediction model structure was modified to evaluate the effectiveness of VMD and sample entropy in reconstructing subsequences, and the effectiveness of adopting a dual-branch structure (processing the reconstructed subsequence and original data sequence, as shown in Fig. 3); (2) The predictive performance of the proposed VMDSE-Tformer model was compared with that of popular and state-of-the-art algorithms such as Informer [37], Transformer [51], Autoformer [52], LSTM [53], LSTNet [15], and TCN [54] with respect to task sequence strength, CPU, and memory requirements.

The evaluation metrics used were Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE), and Relative Standard Error (RSE),

In the experiments, the evaluation metrics used were Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE), and Relative Standard Error (RSE), which were used as comparative indices for analysis. The corresponding formulas for these metrics are:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \times 100\% \tag{38}$$

Table 3
Main parameter configurations of VMDSE-Tformer.

Module parameter	Parameter configuration and description
VMD	Modal component n:14(Task)/9(CPU)/7(Memory)
SE	Number of sub-sequences K:7(Task)/5(CPU)/4(Memory)
VMD-SE-Informer	Multi-Head(8)/Encoder layers(2)/ Decoder layers(1), Dimension of model(512), Feedforward(2048), Dropout(0.1), Learning Rate(0.01), Batch Size(32) MaxPool1d()/AvgPool1d(): kernel_size(3),stride(2),padding(1)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}_i - y_i\right)^2} \tag{39}$$

$$RSE = \frac{100\%}{n}\sqrt{\frac{\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2}}, \tag{40}$$

where $\hat{y}_i$ is the predicted value of the model, y_i is the true data with average $\bar{y}$, and n is the number of samples. MAPE was used to measure the average degree of error in prediction results and evaluate the accuracy and reliability of the prediction model. RMSE measured regression analysis error and the model's predictive ability. RSE is the ratio of actual prediction variance to total observed variance, expressed as a percentage. A smaller RSE generally indicates a better predictive ability of the model.

5.2. The configuration of experimental parameter

The experimental hardware used was Tencent Cloud GPU Computing Type GN7 (8 cores, 32 GB, 5 Mbps, GPU 1× NVIDIA T4). Google Cluster Trace was divided into three categories of sequence types: task request intensity, CPU resource request, and memory resource request. A total of 20,880 samples were extracted based on 2-min statistical results, including 8352 task request intensity sequence samples that were further divided with 5-min statistics. During model training and validation, the dataset was partitioned into training, validation, and test sets in a 7 : 2 : 1 ratio. Table 3 presents the main parameter configurations for VMDSE-Tformer.

5.3. Experimental results and analysis

5.3.1. Ablation experiment analysis

Using the Google Cluster Trace dataset, a set of 20,880 samples was obtained by analyzing the task request intensity sequence at 2-min intervals for the validation experiment. The necessity of the source sequence $AvgPool1d()$ branch and the use of sample entropy for the reconstruction subsequence in VMDSE-Tformer was analyzed.

Let VInformer$_{-i}$ denote the experimental results of VMD modal component i ($i = \{3,5,7\}$) without using the source sequence $AvgPool1d()$ branch. Let V_SInformer$_{-i}$ denote the experimental results of VMD modal component i ($i = \{3,5,7\}$) with the source sequence $AvgPool1d()$ branch, but without using sample entropy for the reconstruction subsequence. VMDSE-Tformer is the proposed prediction model.

When only using VMD and not the $AvgPool1d()$ branch of the source sequence, the $VInformer_{-i}, i \in \{3,5,7\}$ model was tested at prediction requirements of (120,60,60), (90,60,30), and (60,45,20) with 3rd−, 5th−, *and* 7th-order VMD, respectively. MAPE, RMSE, and RSE were analyzed and compared with the proposed prediction model, as shown in Table 4. Fig. 12 shows the improvement of the proposed prediction model.

As shown in Fig. 12, compared to the VInformer$_{-i}$ structure without using the $AvgPool1d()$ branch of the source sequence, the VMDSE-Tformer model showed respective maximum and average improvements of 46.26% and 24.31% in terms of MAPE. In terms of RMSE, these were respectively 14.74% and 4.20%, and in terms of RSE, they were 13.62% and 4.02%. As the number of modal components obtained by VMD increased, there was no significant improvement in RMSE and RSE.

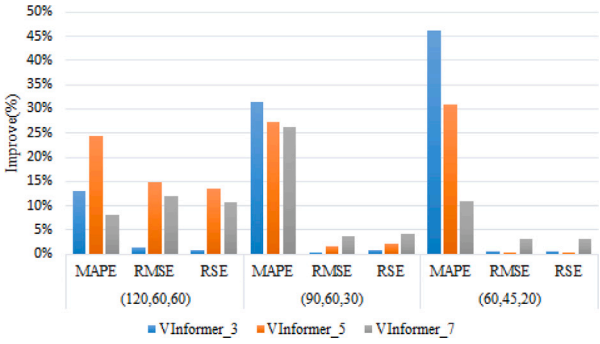


Fig. 12. Improving of VMDSE-Tformer vs. VInformer$_{-i}$, $i \in \{3,5,7\}$.

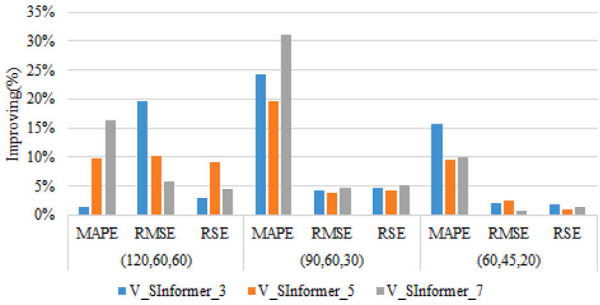


Fig. 13. Improving of VMDSE-Tformer vs. V_SInformer$_{-i}$, $i \in \{3,5,7\}$.

Fig. 13 shows a bar chart of the performance improvement of the V_SInformer$_{-i}, i \in \{3,5,7\}$ prediction model with the source sequence $AvgPool1d()$ branch but without subsequence reconstruction, according to various evaluation metrics. Compared to V_SInformer$_{-i}, i \in \{3,5,7\}$, the proposed prediction model had average improvements of 15.30%, 5.97%, and 3.89% in MAPE, RMSE, and RSE, respectively.

Based on the above comparisons, it can be concluded that VMDSE-Tformer has better prediction performance than VInformer$_{-i}$ and V_SInformer$_{-i}$, which validates the necessity and effectiveness of using VMD, subsequence reconstruction with sample entropy, and the $AvgPool1d()$ branch of the source sequence.

5.3.2. Validation experiment and results

To evaluate the effectiveness of the VMDSE-Tformer prediction model, three resource data sequences were used: the task scheduling intensity sequence, CPU resource request sequence, and memory resource request sequence from the Google Cluster Trace dataset (with a statistical time window of 2 min), with different lengths of input sequence, token input sequences, and prediction lengths as validation targets. The performance of VMDSE-Tformer was compared with state-of-the-art time-series prediction models such as Informer [37], Transformer [51], Autoformer [52], LSTM [53], LSTNet [15], and TCN [54], as summarized in Table 5.

During the experiments, the VMDSE-Tformer model was configured as shown in Table 3, where multidimensional data sequences were used as inputs, and a one-dimensional target sequence was output for prediction, while the comparison prediction models took one-dimensional target sequences as both inputs and outputs.

As demonstrated by the experimental results in Table 5, under different input and prediction sequence lengths, in terms of MAPE, RMSE, and RSE, VMDSE-Tformer achieved the lowest values in 25 out of 27 data points, while Informer achieved the lowest values in

Table 4
Results with different prediction lengths for workloads.

Models		VInformer$_{-i}$			V_SInformer$_{-i}$			VMDSE-
Metric		VMD-3	VMD-5	VMD-7	VMD-3	VMD-5	VMD-7	Tformer
(120,60,60)	MAPE (%)	4.4243	5.0934	4.1871	3.9051	4.2620	4.6001	**3.8475**
	RMSE	0.5635	0.6527	0.6317	0.6929	0.6200	0.5907	**0.5565**
	RSE	1.0161	1.1652	1.1253	1.0370	1.1069	1.0546	**1.0065**
(90,60,30)	MAPE (%)	4.3847	4.1407	4.0817	3.9704	3.7427	4.3697	**3.0091**
	RMSE	0.5612	0.5685	0.5818	0.5843	0.5817	0.5875	**0.5596**
	RSE	1.0039	1.0170	1.0408	1.0453	1.0406	1.0510	**0.9965**
(60,45,20)	MAPE (%)	8.4349	6.5674	5.0914	5.3807	5.0107	5.0303	**4.5332**
	RMSE	0.5630	0.5616	0.5777	0.5710	0.5732	0.5637	**0.5593**
	RSE	1.0072	1.0047	1.0335	1.0195	1.0113	1.0163	**1.0011**
Count		0	0	0	0	0	0	**9**

Table 5
Forecasting results with different methods on tasks-loads.

Metric		VMDSE-Tformer	Transformer	Informer	Autoformer	LSTM	LSTNet	TCN
(120,60,60)	MAPE (%)	**3.848**	4.856	5.331	5.570	5.938	7.854	6.451
	RMSE	**0.557**	0.742	0.564	0.596	0.784	0.925	0.867
	RSE	1.007	1.325	**1.006**	1.064	1.363	1.463	1.392
(120,60,30)	MAPE (%)	**3.005**	4.031	4.850	4.847	6.683	7.956	7.746
	RMSE	**0.557**	0.743	0.598	0.586	0.836	0.918	0.875
	RSE	**0.996**	1.330	1.070	1.049	1.472	1.486	1.483
(120,60,15)	MAPE (%)	3.003	3.616	**2.931**	6.165	7.462	7.673	7.564
	RMSE	**0.542**	0.588	0.553	0.587	0.636	0.762	0.740
	RSE	**0.983**	1.054	0.991	1.053	1.112	1.227	1.198
(90,60,45)	MAPE (%)	**3.761**	4.546	4.340	5.289	6.127	7.485	6.834
	RMSE	**0.552**	0.592	0.562	0.594	0.619	0.721	0.708
	RSE	**0.995**	1.058	1.005	1.063	1.187	1.389	1.309
(90,60,30)	MAPE (%)	**3.009**	4.566	3.715	7.142	7.597	8.876	8.775
	RMSE	**0.559**	0.574	0.564	0.585	0.743	0.788	0.771
	RSE	**0.997**	1.026	1.008	1.046	1.173	1.293	1.225
(90,60,15)	MAPE (%)	**3.199**	4.517	3.426	4.423	5.295	8.129	7.780
	RMSE	**0.533**	0.631	0.555	0.584	0.764	0.772	0.769
	RSE	**0.989**	1.132	0.995	1.048	1.297	1.366	1.313
(60,45,30)	MAPE (%)	**4.012**	4.901	6.767	10.477	9.752	10.896	10.559
	RMSE	**0.549**	0.567	0.577	0.588	0.692	0.882	0.807
	RSE	**0.971**	1.014	1.032	1.052	1.117	1.473	1.329
(60,45,20)	MAPE (%)	**4.533**	5.947	8.899	11.496	11.872	12.771	12.081
	RMSE	**0.559**	0.593	0.560	0.587	0.832	0.961	0.909
	RSE	**1.001**	1.060	1.003	1.051	1.271	1.399	1.348
(60,45,10)	MAPE (%)	**4.418**	6.612	6.042	8.511	9.284	10.663	9.985
	RMSE	**0.545**	0.562	0.551	0.582	0.783	0.942	0.878
	RSE	**1.001**	1.008	0.988	1.044	1.224	1.391	1.322
Count		**25**	0	**2**	0	0	0	0

the other two. Due to the high randomness of the task sequence, weak periodicity, time point characteristics, and coherence features, the performance of all models was poor when predicting short-term task intensity. For example, when predicting with parameters (60,45,20), the MAPE of LSTNet was the worst, at 12.771, while VMDSE-Tformer had a MAPE of 4.533%.

To evaluate the overall performance of each model under different parameters when predicting task sequence (tasks-loads) intensity, the heat maps of each evaluation metric are shown in Figs. 14–16, where a darker color indicates a smaller value, *i.e.*, higher accuracy.

Figs. 14–16 show the heatmap results of MAPE, RMSE, and RSE under various prediction length targets for task sequence intensity, CPU, and RAM resource demands for each model. The color depth of a given region of VMD-SE-Informer is generally the deepest among all models, indicating the best overall performance, followed by Informer and Autoformer. Boxplots in Figs. 17–19 display the percentage improvement of prediction performance on various statistical indicators relative to other models across different prediction datasets.

As shown in Fig. 17, VMDSE-Tformer maintains an absolute advantage over LSTNet and TCN for MAPE, RMSE, and RSE. Its MAPE result has been improved by an average of 59.65%, with the highest reaching 66.10%, compared to LSTNet. In terms of RMSE and RSE, the improvement of VMDSE-Tformer compared to Informer and Autoformer is not significant, with respective average improvement rates of (2.54%, 1.60%) and (6.35%, 5.59%). Therefore, VMDSE-Tformer performs best overall at predicting task intensity. Its prediction ability is shown in Fig. 17 as the circular line indicating the average percentage improvement over different prediction targets.

When predicting different targets for CPU and RAM resource demands, the boxplots and corresponding line charts depicting the distribution and average improvement of prediction accuracy for various prediction models are presented in Figs. 18 and 19, respectively, from which it is evident that VMDSE-Tformer performs better than the other models in terms of improvement in prediction accuracy for both CPU and RAM resource demands.

Figs. 18 and 19 present the distributions of improvement percentages in the predictive ability of VMD-SE-Informer compared to other prediction models for CPU and RAM resource demand data sequences, respectively. Our model shows a significantly higher percentage improvement over LSTNet and TCN for all comparisons and timestamps. Furthermore, for most comparisons and timestamps, it achieves a good prediction accuracy improvement over Transformer and Autoformer.

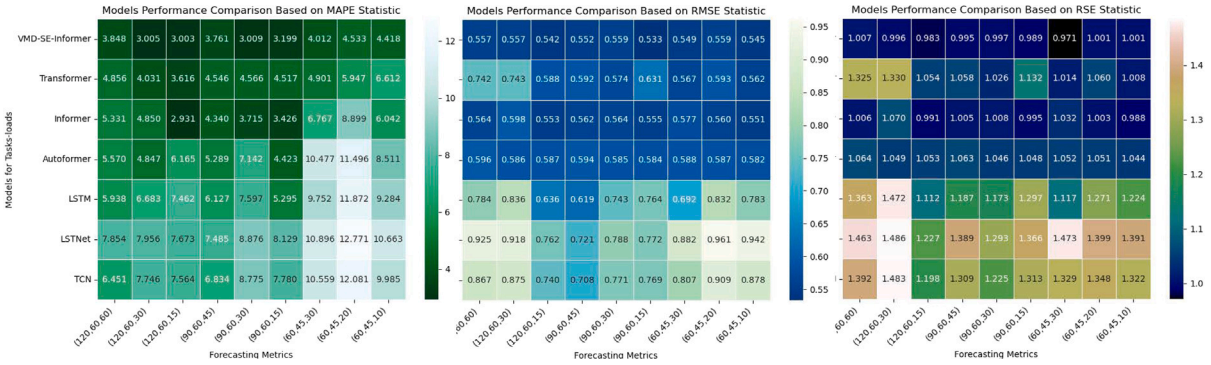


Fig. 14. Heatmap of statistical indicators for task sequence intensity under different prediction length targets for each model.

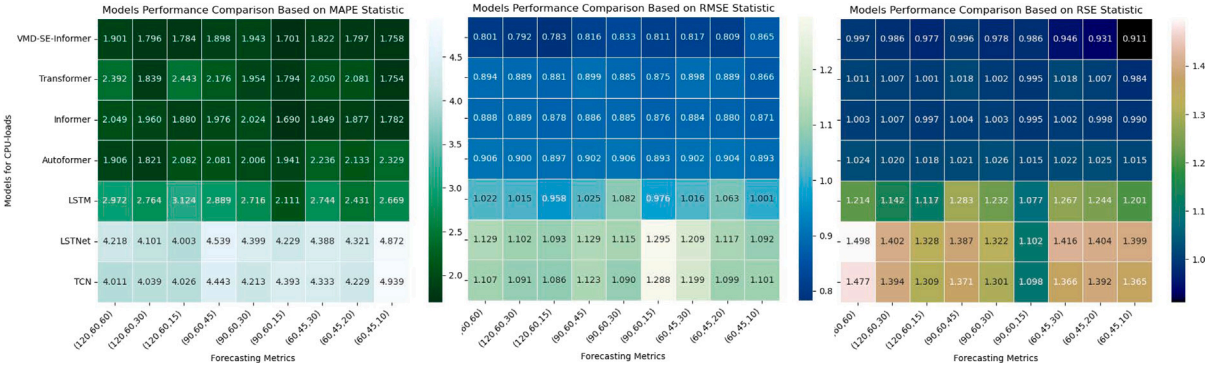


Fig. 15. Heatmap of statistical indicators for CPU resource demand under different prediction length targets for each model.

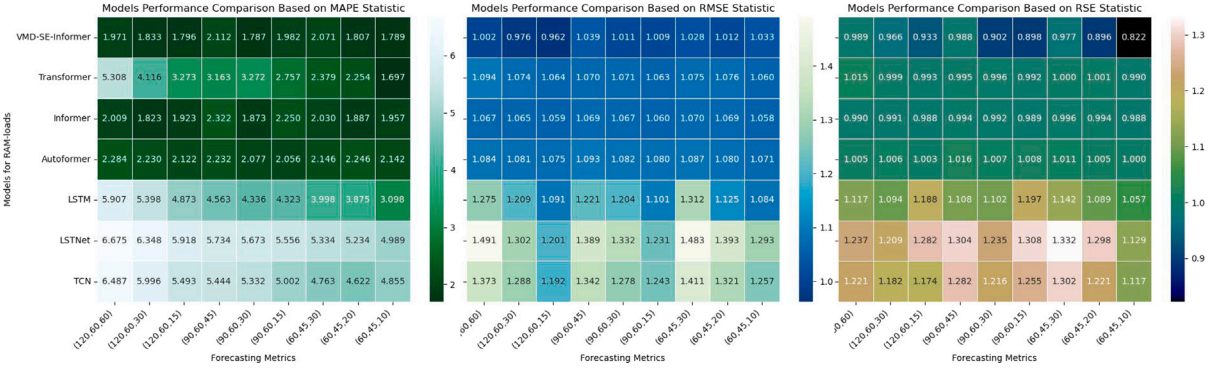


Fig. 16. Heatmap of statistical indicators for RAM resource demand under different prediction length targets for each model.

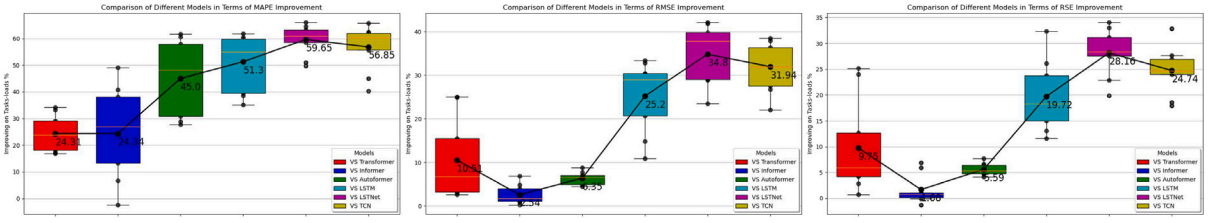


Fig. 17. Boxplot of performance improvement on various statistical indicators for task sequence intensity under different prediction length targets for each model.

Overall, VMDSE-Tformer shows a prediction accuracy improvement of 2.5% to 7.67% over Informer.

Fig. 20 presents the 3D scatter plot of predicted and ground-truth values of VMDSE-Tformer when predicting task sequence intensity parameter indicators (SQ, SL, PL):(120, 60, 30).

From the color distribution of the scatter plot in the 3D results, it can be observed that the differences between predicted and actual values are mainly distributed around zero, with some outliers diffused throughout the plot. Outliers mainly occur due to the model's inability to predict exceptional peak values resulting from sudden and massive

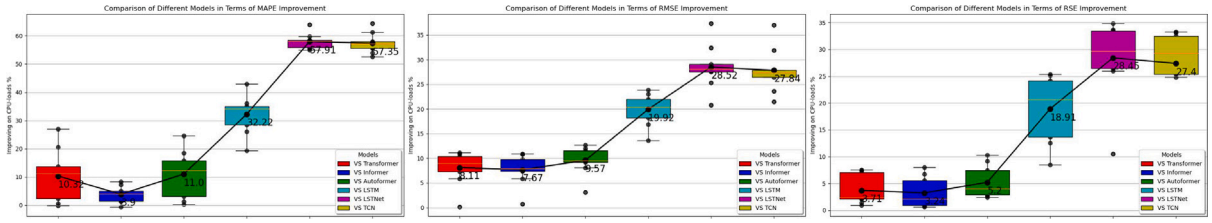


Fig. 18. Boxplot of performance improvement on various statistical indicators for CPU resource demand under different prediction length targets for each model.

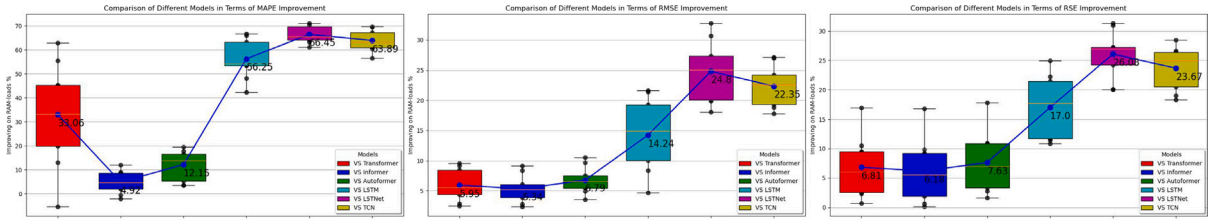


Fig. 19. Boxplot of performance improvement on various statistical indicators for RAM resource demand under different prediction length targets for each model.

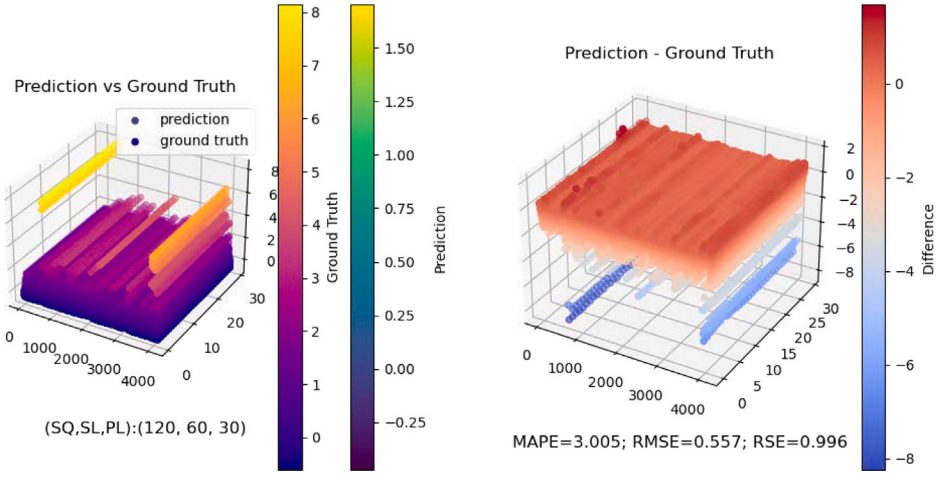


Fig. 20. The 3D scatter for task sequence intensity predictions by VMDSE-Tformer with parameters (120,60,30). SQ is the input sequence length of the encoder. SL is the token length of the decoder. PL is the prediction sequence length.

task scheduling. However, the predicted results generally satisfy the need to predict cloud resource scheduling requests.

Comparing the results of predicted task sequence intensity, CPU, and RAM resource demands with other models, VMDSE-Tformer exhibits the best performance. The distribution of the percentage improvement in overall prediction ability shows that its improvement effect is significantly better and more stable than that of LSTNet and TCN, followed by LSTM and Transformer. Even for Informer and Autoformer, the average percentage improvement is only around 4%.

Given the comprehensive analysis of experimental results, the proposed VMDSE-Tformer is an efficient and accurate prediction model for cloud resource demand. Compared with models such as LSTNet, TCN, LSTM, Transformer, Informer, and Autoformer, it shows substantial advantages in predicting task sequence intensity, as well as CPU and RAM resource demands. Its percentage improvement in predictive ability is higher than that of other models at various lengths of time, with particularly more noticeable and consistent improvement compared to LSTNet and TCN, while the model cannot fully predict abnormal peaks caused by sudden large-scale task scheduling. In this case, we can transfer the prediction task to classification one [55,56]. Overall, it meets the prediction needs of cloud data centers for cloud resource scheduling requests, and hence can be an effective prediction model, providing decision-making support in practical cloud resource scheduling.

6. Conclusion and future work

The prediction model for cloud resource demand plays a vital role in the allocation and scheduling of resources in cloud data centers. By relying on its ability to forecast cloud resource demand, it can help to better understand and analyze the changes and trends in resource usage, leading to optimal resource planning that ensures sufficient availability of resources and avoids shortages or waste. As a result, it can contribute to resource utilization improvement, economic benefits, and ultimately, customer satisfaction, by enabling quicker and more effective responses to customer demands.

We proposed a prediction model for cloud resource demand, VMDSE-Tformer, based on sample entropy optimization and VMD with a selective enhanced attention framework. The model will improve the processing of univariate cloud resource load time series, covering multivariate latent factors, to enhance the accuracy and reliability of load prediction. It decomposes time-series data through VMD, and reconstructs the subsequence set using sample entropy calculations, so as to extract latent features from the data sequence. A Transformer-like framework based on multi-head self-attention is subsequently employed for deep learning on each component sequence to obtain encoded representations of multiple associated feature information. In experiments, VMDSE-Tformer outperformed several state-of-the-art

models at predicting task sequence intensity, CPU, and RAM resource demands, providing stable and significant improvements in predictive ability. The average improvements of the proposed model on the MAPE metric are as follows: 59.65% for Workload, 57.91% for Cpuload, and 66.45% for RAMload, as shown in Figs. 17 to 19. It demonstrates absolute advantages in various statistical metrics, especially in terms of MAPE, where it achieves an average improvement of around 60% compared to LSTNet. In predicting CPU and RAM resource demand data, VMDSE-Tformer had a more accurate predictive ability than other models, with a stable improvement effect. In most cases, the model achieved high accuracy and robustness, effectively meeting the prediction needs of cloud data centers for resource scheduling requests. Overall, VMDSE-Tformer can contribute to more efficient and effective resource management. In the future, we plan to improve the predictive performance of the model and explore its applicability in more complex and dynamic cloud environments.

CRediT authorship contribution statement

Jiaxian Zhu: Writing – original draft, Methodology. **Weihua Bai:** Software, Methodology. **Jialing Zhao:** Data curation. **Liyun Zuo:** Formal analysis. **Teng Zhou:** Funding acquisition, Conceptualization. **Keqin Li:** Supervision.

Declaration of competing interest

The authors declare that they have no conflict of interest.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by the 2023 Guangdong Scientific Research Platform and Projects for the Higher-educational Institution and Education Science Planning Scheme, Guangdong Basic and Applied Basic Research Foundation, China (No. 2023A1515012874, 2022A1515011590, 2021A1515012302, 2022A1515011978), the Special Fund for Science and Technology of Guangdong Province, China under Grant (No. 2021S0053), the National Natural Science Foundation of China (No. 82020108016, 61902232), Key Scientific Research Project of Universities in Guangdong Province, China (No. 2020ZDZX3028, 2022ZDZX1007), and Guangdong Provincial Science and Technology Plan Project, China (No. STKJ202209003).

References

[1] M. Shojafar, N. Cordeschi, E. Baccarelli, Energy-efficient adaptive resource management for real-time vehicular cloud services, IEEE Trans. Cloud Comput. 7 (1) (2016) 196–209.

[2] D. Fernández-Cerero, A. Fernández-Montes, A. Jakóbik, J. Kołodziej, M. Toro, Score: Simulator for cloud optimization of resources and energy consumption, Simul. Model. Pract. Theory 82 (2018) 160–173.

[3] D.-M. Bui, Y. Yoon, E.-N. Huh, S. Jun, S. Lee, Energy efficiency for cloud computing system based on predictive optimization, J. Parallel Distrib. Comput. 102 (2017) 103–114.

[4] J. Kumar, D. Saxena, A. Singh, A. Mohan, Biphase adaptive learning-based neural network model for cloud datacenter workload forecasting, Soft Comput. 24 (2020) 14593–14610.

[5] D. Saxena, A. Singh, A proactive autoscaling and energy-efficient vm allocation framework using online multi-resource neural network for cloud data center, Neurocomputing 426 (2021) 248–264.

[6] J. Bi, S. Li, H. Yuan, M. Zhou, Integrated deep learning method for workload and resource prediction in cloud systems, Neurocomputing 424 (2021) 35–48.

[7] W. Bai, J. Zhu, S. Huang, H. Zhang, A queue waiting cost-aware control model for large scale heterogeneous cloud datacenter, IEEE Trans. Cloud Comput. 10 (2) (2022) 849–862.

[8] J. Gao, H. Wang, H. Shen, Machine learning based workload prediction in cloud computing, in: 2020 29th International Conference on Computer Communications and Networks (ICCCN), IEEE, 2020, pp. 1–9.

[9] M. Al-Asaly, M. Bencherif, A. Alsanad, M. Hassan, A deep learning-based resource usage prediction model for resource provisioning in an autonomic cloud computing environment, Neural Comput. Appl. (2021) 1–18.

[10] D. AbdElminaam, A. Toony, M. Taha, Resource allocation in the cloud environment based on quantum genetic algorithm using kalman filter with anfis, IJCSNS 20 (10) (2020) 10.

[11] T. Gyeera, A. Simons, M. Stannett, Kalman filter based prediction and forecasting of cloud server kpis, IEEE Trans. Serv. Comput. (2022).

[12] H. Mehdi, Z. Pooranian, P. Vinueza Naranjo, Cloud traffic prediction based on fuzzy arima model with low dependence on historical data, Trans. Emerg. Telecommun. Technol. 33 (3) (2022) e3731.

[13] B. Huang, Y. Song, Z. Cui, H. Dou, D. Jiang, T. Zhou, J. Qin, Gravitational search algorithm-extreme learning machine for covid-19 active cases forecasting, IET Softw. 17 (4) (2023) 554–565.

[14] W. Xu, J. Liu, J. Yan, J. Yang, H. Liu, T. Zhou, Dynamic spatiotemporal graph wavelet network for traffic flow prediction, IEEE Internet Things J. (2023).

[15] G. Lai, W.-C. Chang, Y. Yang, H. Liu, Modeling long-and short-term temporal patterns with deep neural networks, in: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018, pp. 95–104.

[16] K. Anupama, B. Shivakumar, R. Nagaraja, Resource utilization prediction in cloud computing using hybrid model, Int. J. Adv. Comput. Sci. Appl. 12 (4) (2021).

[17] Y. Han, Y. Jing, K. Li, G. Dimirovski, Network traffic prediction using variational mode decomposition and multi-reservoirs echo state network, Ieee Access 7 (2019) 138364–138377.

[18] P. Yazdanian, S. Sharifian, E2lg: a multiscale ensemble of lstm/gan deep learning architecture for multistep-ahead cloud workload prediction, J. Supercomput. 77 (2021) 11052–11082.

[19] J. Dogani, F. Khunjush, M. Seydali, Host load prediction in cloud computing with discrete wavelet transformation (dwt) and bidirectional gated recurrent unit (bigru) network, Comput. Commun. 198 (2023) 157–174.

[20] P. Osypanka, P. Nawrocki, Qos-aware cloud resource prediction for computing services, IEEE Trans. Serv. Comput. 16 (2) (2022) 1346–1357.

[21] E. Patel, D. Kushwaha, An integrated deep learning prediction approach for efficient modelling of host load patterns in cloud computing, J. Grid Comput. 21 (1) (2023) 5.

[22] S. Ouhame, Y. Hadi, A. Ullah, An efficient forecasting approach for resource utilization in cloud data center using cnn-lstm model, Neural Comput. Appl. 33 (2021) 10043–10055.

[23] A. Ma, Y. Gao, L. Huang, B. Zhang, Improved differential search algorithm based dynamic resource allocation approach for cloud application, Neural Comput. Appl. 31 (2019) 3431–3442.

[24] N. Tran, T. Nguyen, B. Nguyen, G. Nguyen, A multivariate fuzzy time series resource forecast model for clouds using lstm and data correlation analysis, Procedia Comput. Sci. 126 (2018) 636–645.

[25] W. Iqbal, J. Berral, D. Carrera, et al., Adaptive sliding windows for improved estimation of data center resource utilization, Future Gener. Comput. Syst. 104 (2020) 212–224.

[26] M. Hilman, M. Rodriguez, R. Buyya, Task runtime prediction in scientific workflows using an online incremental learning approach, in: 2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC), IEEE, 2018, pp. 93–102.

[27] P. Nawrocki, M. Grzywacz, B. Sniezynski, Adaptive resource planning for cloud-based services using machine learning, J. Parallel Distrib. Comput. 152 (2021) 88–97.

[28] P. Nawrocki, P. Osypanka, B. Posluszny, Data-driven adaptive prediction of cloud resource usage, J. Grid Comput. 21 (1) (2023) 6.

[29] J. Li, B. Yang, H. Li, Y. Wang, C. Qi, Y. Liu, Dtdr–alstm: Extracting dynamic time-delays to reconstruct multivariate data for improving attention-based lstm industrial time series prediction models, Knowl.-Based Syst. 211 (2021) 106508.

[30] T.-H. Cheung, D.-Y. Yeung, Modals: Modality-agnostic automated data augmentation in the latent space, in: International Conference on Learning Representations (ICLR), 2020.

[31] Y. Kang, R. Hyndman, F. Li, Gratis: Generating time series with diverse and controllable characteristics, Stat. Anal. Data Min.: ASA Data Sci. J. 13 (4) (2020) 354–376.

[32] K. Benidis, S. Rangapuram, V. Flunkert, Y. Wang, D. Maddix, C. Turkmen, J. Gasthaus, M. Bohlke-Schneider, D. Salinas, L. Stella, et al., Deep learning for time series forecasting: Tutorial and literature survey, ACM Comput. Surv. 55 (6) (2022) 1–36.

[33] S. Arbat, V. Jayakumar, J. Lee, W. Wang, I. Kim, Wasserstein adversarial transformer for cloud workload prediction, in: Proceedings of the AAAI Conference on Artificial Intelligence, 36, 2022, pp. 12433–12439.

[34] J. Yoon, D. Jarrett, M. Van der Schaar, Time-series generative adversarial networks, Adv. Neural Inf. Process. Syst. 32 (2019).

[35] E. Fons, P. Dawson, X.-j. Zeng, J. Keane, A. Iosifidis, Adaptive weighting scheme for automatic time-series data augmentation, 2021, arXiv preprint arXiv:2102.08310.

[36] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao, X. Wang, H. Xu, Time series data augmentation for deep learning: A survey, in: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI), International Joint Conferences on Artificial Intelligence Organization, 2021.
[37] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: Beyond efficient transformer for long sequence time-series forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, 35, 2021, pp. 11106–11115.
[38] S. Weichwald, M. Jakobsen, P. Mogensen, L. Petersen, N. Thams, G. Varando, Causal structure learning from time series: Large regression coefficients may predict causal links better in practice than small p-values, PMLR, 2020, pp. 27–36.
[39] M. Gong, K. Zhang, B. Schölkopf, C. Glymour, D. Tao, Causal discovery from temporally aggregated time series, in: Uncertainty in Artificial Intelligence: Proceedings of the. Conference. Conference on Uncertainty in Artificial Intelligence, NIH Public Acces, 2017.
[40] W. Yao, Y. Sun, A. Ho, C. Sun, K. Zhang, Learning temporally causal latent processes from general temporal data, in: International Conference on Learning Representations (ICLR), 2021.
[41] K. Dragomiretskiy, D. Zosso, Variational mode decomposition, IEEE Trans. Signal Process. 62 (3) (2013) 531–544.
[42] N. Ur Rehman, H. Aftab, Multivariate variational mode decomposition, IEEE Trans. Signal Process. 67 (23) (2019) 6039–6052.
[43] J. Richman, J. Moorman, Physiological time-series analysis using approximate entropy and sample entropy, Am. J. Physiol.-Heart Circul. Physiol. 278 (6) (2000) H2039–H2049.
[44] N. Liu, F. Li, D. Wang, J. Gao, Z. Xu, Ground-roll separation and attenuation using curvelet-based multichannel variational mode decomposition, IEEE Trans. Geosci. Remote Sens. 60 (2021) 1–14.
[45] J. Yentes, N. Hunt, K. Schmid, J. Kaipust, D. McGrath, N. Stergiou, The appropriate use of approximate entropy and sample entropy with short data sets, Ann. Biomed. Eng. 41 (2013) 349–365.
[46] J. Jiang, S. Tian, Y. Tian, Y. Zhou, C. Hu, Transient abnormal signal acquisition system based on approximate entropy and sample entropy, Rev. Sci. Instrum. 93 (4) (2022).
[47] D. Yin, B. Zhang, J. Yan, Y. Luo, T. Zhou, J. Qin, Cownet: A correlation weighted network for geological hazard detection, Knowl.-Based Syst. (2023) 1–10.
[48] B. Huang, H. Dou, Y. Luo, J. Li, J. Wang, T. Zhou, Adaptive spatiotemporal transformer graph network for traffic flow forecasting by iot loop detectors, IEEE Internet Things J. (2022).
[49] Y. Luo, Q. Huang, J. Ling, K. Lin, T. Zhou, Local and global knowledge distillation with direction-enhanced contrastive learning for single-image deraining, Knowl.-Based Syst. 268 (2023) 110480.
[50] C. Reiss, J. Wilkes, J. Hellerstein, Google Cluster-Usage Traces: Format+ Schema, White Paper 1, Google Inc, 2011, pp. 1–14.
[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).
[52] H. Wu, J. Xu, J. Wang, M. Long, Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting, Adv. Neural Inf. Process. Syst. 34 (2021) 22419–22430.
[53] K. Kawakami, Supervised Sequence Labelling with Recurrent Neural Networks (Ph.D. thesis), Technical University of Munich, 2008.
[54] S. Bai, J. Kolter, V. Koltun, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018, arXiv preprint arXiv:1803.01271.
[55] T. Zhou, H. Dou, J. Tan, Y. Song, F. Wang, J. Wang, Small dataset solves big problem: an outlier-insensitive binary classifier for inhibitory potency prediction, Knowl.-Based Syst. (2022).
[56] T. Quan, Y. Yuan, Y. Luo, Y. Song, T. Zhou, J. Wang, From regression to classification: Fuzzy multi-kernel subspace learning for robust prediction and drug screening, IEEE Trans. Ind. Inform. (2023).