Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.064880

ARTICLE

SDVformer: A Resource Prediction Method for Cloud Computing Systems

Shui Liu[1,2], Ke Xiong[1,2,*], Yeshen Li[1,2], Zhifei Zhang[1,2,*], Yu Zhang[3] and Pingyi Fan[4]

[1]Engineering Research Center of Network Management Technology for High Speed Railway of Ministry of Education, School of Computer Science and Technology, Beijing Jiaotong University, Beijing, 100044, China

[2]Collaborative Innovation Center of Railway Traffic Safety, National Engineering Research Center of Advanced Network Technologies, Beijing Jiaotong University, Beijing, 100044, China

[3]State Grid Energy Research Institute Co., Ltd., Beijing, 102209, China

[4]Department of Electronic Engineering, Tsinghua University, Beijing, 100084, China

*Corresponding Authors: Ke Xiong. Email: kxiong@bjtu.edu.cn; Zhifei Zhang. Email: zhfzhang@bjtu.edu.cn

Received: 26 February 2025; Accepted: 22 May 2025; Published: 30 July 2025

ABSTRACT: Accurate prediction of cloud resource utilization is critical. It helps improve service quality while avoiding resource waste and shortages. However, the time series of resource usage in cloud computing systems often exhibit multidimensionality, nonlinearity, and high volatility, making the high-precision prediction of resource utilization a complex and challenging task. At present, cloud computing resource prediction methods include traditional statistical models, hybrid approaches combining machine learning and classical models, and deep learning techniques. Traditional statistical methods struggle with nonlinear predictions, hybrid methods face challenges in feature extraction and long-term dependencies, and deep learning methods incur high computational costs. The above methods are insufficient to achieve high-precision resource prediction in cloud computing systems. Therefore, we propose a new time series prediction model, called SDVformer, which is based on the Informer model by integrating the Savitzky-Golay (SG) filters, a novel Discrete-Variation Self-Attention (DVSA) mechanism, and a type-aware mixture of experts (T-MOE) framework. The SG filter is designed to reduce noise and enhance the feature representation of input data. The DVSA mechanism is proposed to optimize the selection of critical features to reduce computational complexity. The T-MOE framework is designed to adjust the model structure based on different resource characteristics, thereby improving prediction accuracy and adaptability. Experimental results show that our proposed SDVformer significantly outperforms baseline models, including Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Informer in terms of prediction precision, on both the Alibaba public dataset and the dataset collected by Beijing Jiaotong University (BJTU). Particularly compared with the Informer model, the average Mean Squared Error (MSE) of SDVformer decreases by about 80%, fully demonstrating its advantages in complex time series prediction tasks in cloud computing systems.

KEYWORDS: Cloud computing; time series prediction; DVSA; SG filter; T-MOE

1 Introduction

The rapid advancement of Internet of Things (IoT) technology has led to an exponential growth in the number of connected devices and the volume of data [1]. In this context, efficiently storing, computing, and analyzing massive amounts of data has become a challenging issue [2]. Cloud computing is a distributed model that provides computing resources and services by uploading the computational tasks to the cloud, with its core design principles focusing on resource virtualization and dynamic scheduling. Users can access computing, storage, and network resources on a pay-as-you-go basis without physical hardware, which effectively reduces capital expenditure (CAPEX) and operational costs, while significantly improving scalability and resource utilization efficiency [3]. With its powerful capabilities, elastic storage, and on-demand resource allocation, cloud computing provides critical support for the development of IoT technology and has gradually become a research hotspot in both academics and industry.

However, cloud computing's powerful capabilities rely on the rational allocation and efficient utilization of a substantial amount of underlying resources, such as CPUs and memory. Therefore, accurately predicting the utilization of these underlying resources over a future period is of critical importance. Accurate resource utilization forecasting enables the optimization of resource scheduling and allocation in cloud computing systems, maximizing the utilization of underlying resources, and minimizing resource idleness and waste, thereby improving overall system efficiency and effectively reducing operational costs [4].

Cloud computing resources exhibit time series characteristics, as they are continuously recorded and dynamically change over time. Therefore, cloud resource utilization prediction essentially falls within the domain of time series forecasting. With the rapid growth in computational demands, time series forecasting has attracted significant research interest in recent years. Traditional forecasting methods, such as the Autoregressive (AR) model [5] and Seasonal Autoregressive Integrated Moving Average (SARIMA) model, have limitations in handling complex, nonlinear, large-scale data, resulting in lower prediction accuracy. Compared to traditional methods, Recurrent Neural Networks (RNNs) [6] have gained widespread application in time series forecasting, demonstrating excellent capability for processing complex data and capturing nonlinear relationships. Furthermore, RNN variants, such as Long Short-Term Memory (LSTM) [7] and Bidirectional LSTM (Bi-LSTM) [8], have been extensively applied in time series forecasting, further improving accuracy and effectiveness.

Nonetheless, LSTM and Bi-LSTM have limitations, such as high computational complexity and limited parallelization, which make them less efficient for long sequences or high-dimensional data. To overcome these limitations, the Informer model is proposed. With its self-attention mechanism, the Informer efficiently captures global dependencies within a sequence, regardless of distance, overcoming the information loss often seen in RNNs. Additionally, multi-head attention focuses on different sequence features, enhancing the model's understanding of complex dependencies. Positional encoding introduces sequential information, making Informer suitable for time series data. These modules make Informer more effective for long-time series prediction than RNN and its variants.

Although Informer demonstrates excellent performance in long sequence time series prediction, it still faces limitations such as insufficient feature extraction capability, sensitivity to noise, and limited generalization ability. These issues result in strong performance on specific data types but suboptimal results on others.

Resource usage time series in cloud computing environments often exhibit multidimensionality, nonlinearity, and high volatility, making them more challenging to predict compared to general time series data [9]. If one directly applies Informer to predict cloud resource usage amounts, including CPU, memory, network bandwidth, and storage, the results may not be accurate due to the limitations of Informer mentioned previously. To address this issue, a novel prediction architecture called SDVformer is proposed for cloud computing systems based on Informer, which inherits the advantages of Informer but at the same time avoids its shortcomings.

For clarity, the primary contributions of this paper are summarized as follows:

- **SDVformer:** We propose a novel prediction architecture called SDVformer, a version tailored for time series forecasting of cloud computing resources, based on the Informer model, integrating our newly designed Discrete-Variation Self-Attention (DVSA), the classic Savitzky-Golay (SG) Filter, and our newly presented type-aware Mixture of Experts Model (T-MOE).

- **DVSA:** We propose a new self-attention mechanism, called DVSA, which is designed to efficiently extract attention features while preserving model performance. Compared to Informer's ProbSparse self-attention, DVSA employs a self-designed sparsity measure to capture the most important queries. Experiments in Section 4.3.2 show that DVSA outperforms the existing Informer's ProbSparse self-attention.
- **SG Filter Mechanism:** Cloud computing resource data often exhibit large fluctuations over short periods, which may introduce noise. As is known, using filters for denoising can effectively reduce noise and improve the quality of the data. Nevertheless, to the best of our knowledge, integrating filters into transformer models has not been examined in previous works. Therefore, in this work, we apply the SG filter to the proposed SDVformer to enhance the model's ability to capture meaningful features within the time series data.
- **T-MOE:** Cloud computing resource data are diverse (e.g., CPU and memory), with each kind of data having its unique characteristics. This diversity may cause a prediction method to perform well when processing one kind of data but poorly when handling another kind. To ensure that the prediction method performs well across different types of data within a single cloud computing system, we propose a T-MOE framework. This framework tailors expert systems for different resource types, allowing the model to effectively handle time series predictions for various cloud computing resources. Compared with traditional MOE, T-MOE introduces a novel approach that adapts expert systems to the specific characteristics of each resource type, thereby enhancing model performance across diverse cloud computing resources.
- Experimental results indicate that SDVformer, the proposed model, achieves substantially higher prediction accuracy compared to baseline models such as RNN, LSTM, and Informer. This improvement is consistently observed across both the Alibaba public dataset and the dataset collected by Beijing Jiaotong University (BJTU). Notably, when compared to the Informer model, SDVformer reduces the average MSE by approximately 80%, highlighting its effectiveness in handling complex time series prediction tasks within cloud computing systems.

2 Related Work

Cloud computing resource prediction methods can be broadly categorized into three types: traditional statistical methods, hybrid approaches combining machine learning with classical models, and deep learning-based prediction methods. Traditional statistical methods apply time-series models for purely linear predictions. Hybrid machine learning and classical models perform better in complex data structures, while deep learning methods excel in multi-step prediction and nonlinear feature extraction.

2.1 Traditional Statistical Methods

Zhang et al. [10] proposed a hybrid method that combines the ARIMA model with an Artificial Neural Network (ANN) approach. The ARIMA model handles the linear components, while the ANN captures nonlinear elements to enhance prediction accuracy. Experiments show that this hybrid model outperforms the standalone use of either ARIMA or ANN across different datasets.

Calheiros et al. [11] used an ARIMA-based workload prediction method to ensure quality assurance, predicting future resource demand to optimize cloud service configuration and significantly improve resource utilization and Quality of Service (QoS). Despite the ARIMA model's strong performance in time-series forecasting, it struggles to handle sudden load spikes.

While traditional statistical methods offer high prediction accuracy using time-series models, they are primarily suited for linear and stable workload forecasts. These methods face challenges with sudden, nonlinear loads, often requiring combination with other models to improve prediction outcomes.

2.2 Hybrid Machine Learning and Classical Models

Qiu et al. [12] proposed an Oblique Random Forest (ORF) ensemble method based on least-squares estimation. By partitioning data on oblique hyperplanes, ORF captures the geometric structure of data better than traditional Random Forests, achieving higher prediction accuracy across multiple datasets, though at a higher computational cost.

Mallick et al. [13] applied a combined Naive Bayes and Markov model in a virtualized server environment, using log data to predict system resource usage, helping system administrators anticipate potential resource bottlenecks.

Hu et al. [14] developed a multi-step prediction method using Support Vector Regression (SVR) combined with a Kalman Smoother (KSSVR) that suits complex cloud environments. Experimental results indicate that this method surpasses traditional AR and Backpropagation Neural Network (BPNN) methods in accuracy and stability.

Hybrid approaches enhance nonlinear modeling capability, improving multivariate feature extraction and predictive accuracy, making them suitable for complex data structures with high-dimensional features. However, these methods still face limitations in automatic feature extraction, multi-level nonlinear modeling, and capturing long-term dependencies and complex patterns.

2.3 Deep Learning-Based Prediction Methods

With the increasing complexity of cloud computing environments, deep learning methods have become a focus of research. Duggan et al. [15] designed a Recurrent Neural Network (RNN)-based model for single-step and multi-step CPU utilization prediction in the cloud, where RNN performs well for short-term load forecasting, effectively supporting resource management.

Karim et al. [16] introduced the BHyPreC model, which combines Bidirectional LSTM, LSTM, and GRU units for predicting CPU load in cloud virtual machines. This model extracts historical and future information using multiple deep learning units, significantly improving prediction accuracy, especially in multi-step forecasting.

Vaswani et al. [17] developed the Transformer model, which relies entirely on self-attention mechanisms rather than recursion or convolution. The self-attention mechanism establishes global dependencies between inputs and outputs, enabling parallel processing and high training efficiency. Zhou et al. [18] proposed an efficient Transformer-based time-series prediction model called *Informer*, which incorporates a ProbSparse self-attention mechanism to reduce computational complexity in long-sequence prediction. The informer focuses on essential information through self-attention distillation and accelerates the prediction process with a generative decoder. Experimental results in [18] on various large-scale datasets demonstrate that Informer achieves notable performance improvements in long-sequence prediction tasks.

Deep learning-based prediction methods, particularly Transformer and its variants [19], leverage self-attention mechanisms to capture nonlinear patterns in time-series data, achieving high accuracy and efficiency in complex, multi-step predictions. While these methods are computationally intensive, they are broadly applicable to a variety of complex cloud computing load prediction tasks.

3 Problem Statement and Model Architectures

This section will introduce the SDVformer model in detail. In Section 3.1, we define the application scenarios and goals that the model aims to solve. In Section 3.2, we briefly introduced the basic structure of Informer. In Section 3.3, we presented the structure of SDVformer. Since SDVformer builds upon improvements to Informer, we delve into the three key enhancements of SDVformer over Informer in Section 3.4, covering each improvement's structural design, motivation, and specific functionality.

3.1 Problem Definition

Consider a typical scenario of cloud computing resource time series forecasting, the usage record of a cloud computing resource (such as CPUs, memory, network bandwidth, and storage) associated the just passed N time slots is known, which is defined as the input sequence $\mathcal{X}^t = \left\{x_1^t, \ldots, x_N^t \mid x_i^t \in \mathbb{R}^{d_x}\right\}$, where d_x represents the feature dimension at each time step. Our objective is to predict the usage for the next M time slots $\mathcal{Y}^t = \left\{y_1^t, \ldots, y_M^t \mid y_i^t \in \mathbb{R}^{d_y}\right\}$, where d_y indicates the feature dimension of the prediction output. Consequently, we have:

$$\mathcal{Y}^t = f\left(\mathcal{X}^t\right), \tag{1}$$

where $\mathcal{Y}^t$ represents the predicted output sequence, $\mathcal{X}^t$ denotes the input sequence at time t, and $N, M, d_x, d_y \geq 1$.

3.2 Informer Model

The Informer model primarily consists of an encoder, decoder, and self-attention mechanism, with its overall architecture based on Transformer, but optimized for computational efficiency in long-sequence forecasting. Compared to traditional Transformers, which suffer from high computational complexity when handling long sequences, Informer improves efficiency through structural modifications, making it more applicable for large-scale time-series forecasting tasks.

The core contribution of the Informer model lies in the optimization of the self-attention mechanism. Traditional self-attention mechanisms exhibit quadratic growth in computational cost when processing long sequences, leading to excessive consumption of computational resources. In contrast, Informer introduces ProbSparse Self-Attention, which reduces computational complexity by selecting the most representative attention scores, focusing only on the key information crucial for predictions. This approach not only effectively reduces redundant computations but also maintains temporal dependencies, thereby improving both computational efficiency and prediction accuracy, enabling superior performance when handling long time-series data.

3.3 SDVformer Model

The SDVformer model structure is proposed to address the challenges of forecasting various cloud computing resources in cloud computing systems. The model's core components include a Filter, an Encoder, and a Decoder, with multiple innovative improvements that significantly enhance processing efficiency and prediction accuracy. Furthermore, to achieve precise forecasting across multiple types of cloud resources, the model incorporates a T-MoE mechanism. The structure of the SDVformer model is illustrated in Fig. 1. In this section, we will provide an in-depth analysis of the encoder and decoder structures.

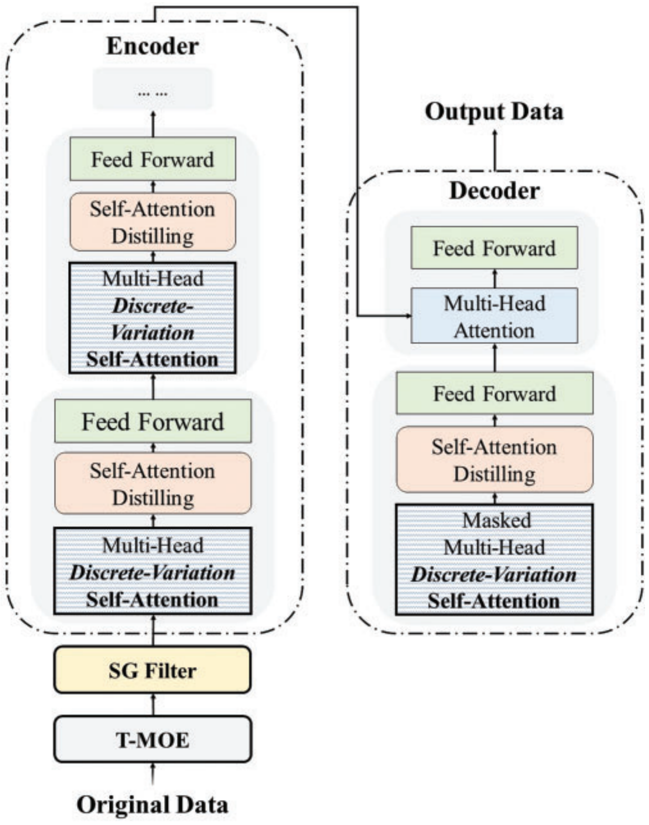

Encoder
... ...
Feed Forward
Self-Attention Distilling
Multi-Head Discrete-Variation Self-Attention
Feed Forward
Self-Attention Distilling
Multi-Head Discrete-Variation Self-Attention
SG Filter
T-MOE
Original Data
Output Data
Decoder
Feed Forward
Multi-Head Attention
Feed Forward
Self-Attention Distilling
Masked Multi-Head Discrete-Variation Self-Attention

Figure 1: Model archtectures: the proposed SDVformer

3.3.1 Encoder

The input data (preprocessed by a filter) is first mapped to a high-dimensional feature space through an embedding layer, with positional encoding added to incorporate sequence order information. Next, the data passes through multiple encoder layers, each containing a DVSA mechanism, which focuses only on the most critical dependencies, thereby reducing computational complexity. Additionally, the Self-Attention Distilling operation refines attention scores to extract dominant attention features, further simplifying computational demands. The multi-head attention mechanism enables the model to capture diverse features, and each layer also includes a Feed-forward Network, Layer Normalization, and Residual Connections to enhance stable information flow and support deep feature extraction. Through these processing steps, the encoder effectively captures long-range dependencies and key features within the time series, providing rich contextual information for the subsequent decoding process.

Self-Attention Distilling: The self-attention distilling operation is introduced to efficiently handle long sequence inputs and reduce computational complexity. This operation first extracts local features from the time series using a 1-D convolution and ELU activation function, forming a focused attention feature map. Next, a max-pooling operation with a stride of 2 is applied in each layer, progressively halving the sequence length. This approach gradually reduces the input dimensions, removes redundant information, and optimizes memory usage. The data is transferred from layer j to layer $j + 1$ as follows:

$$\mathcal{X}_{j+1}^{t} = \text{MaxPool}\left(\text{ELU}\left(\text{Conv1d}\left[\mathcal{X}_{j}^{t}\right]_{AB}\right)\right), \tag{2}$$

where X_j^t represents the input of the j-th layer, and $[\cdot]_{AB}$ denotes the attention block, which includes multi-head DVSA and convolutional operations. Conv1d is a one-dimensional convolution with a kernel size of 3, accompanied by the ELU activation function.

3.3.2 Decoder

The primary function of the decoder is to generate prediction outputs, using a generative inference approach to progressively produce multi-step forecasting results. We adopted a standard decoder structure (referencing Vaswani et al., 2017), consisting of two stacked identical multi-head attention layers. To accelerate inference speed for long sequence predictions, a Generative Inference mechanism is specifically introduced.

Generative Inference: The model uses a "start token" to guide the decoding process in generative inference. This concept has been widely applied in "dynamic decoding" in natural language processing (NLP). Typically, the decoder requires a starting point to generate subsequent outputs; however, in generative inference, this starting point is not a fixed token but rather a segment extracted from the input sequence. For example, if we want to predict the CPU usage of a cloud server for the next 24 h, we can use the known CPU usage data from the previous 48 h as the "start token" for the decoder. This approach enables the decoder to generate all prediction results in a single forward pass, thus avoiding the time-consuming step-by-step process of "dynamic decoding".

Loss Function: We use the MSE loss function to measure the difference between the predicted and target sequences, and this loss is backpropagated from the decoder outputs throughout the entire model.

3.4 Key Machnisms in SDVformer

3.4.1 Discrete-Variation Self-Attention (DVSA)

In the standard self-attention mechanism proposed by Vaswani et al. (2017), the attention computation is defined based on three inputs: Query **Q**, Key **K**, and Value **V**. The computation is performed as a scaled dot-product, which is given by:

$$Attention\left(\mathbf{Q},\mathbf{K},\mathbf{V}\right)=\text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{d}}\right)\mathbf{V}, \tag{3}$$

where $\mathbf{Q}\in\mathbb{R}^{L_Q\times d}$, $\mathbf{K}\in\mathbb{R}^{L_K\times d}$, and $\mathbf{V}\in\mathbb{R}^{L_V\times d}$ represent the Query, Key, and Value matrices, respectively, and d denotes the input dimensionality.

The traditional self-attention mechanism effectively captures global dependencies but is limited by high computational and memory demands. Sparse attention improves efficiency by focusing only on key positions in the input sequence.

In the attention mechanism, a sparsity measure is assigned to each query vector $\mathbf{q}_i$ to evaluate its contribution to the global attention distribution. Specifically, the sparsity measure of the i-th query vector is defined as:

$$Sparsity\left(\mathbf{q}_i,\mathbf{K}\right)=\sum_{j=1}^{L_K}\left(\frac{\mathbf{q}_i\mathbf{k}_j^{\top}}{\sqrt{d}}-\frac{1}{L_K}\sum_{j=1}^{L_K}\frac{\mathbf{q}_i\mathbf{k}_j^{\top}}{\sqrt{d}}\right)^2, \tag{4}$$

where the first term of the absolute value formula represents the dot product between the query $\mathbf{q}_i$ and the key $\mathbf{k}_j$, and the second term represents the arithmetic mean of the dot products between $\mathbf{q}_i$ and all $\mathbf{K}$. This formula draws inspiration from the concept of the variance formula, which is commonly used to measure the degree of fluctuation in data. Here, the computation based on Eq. (4) is used to quantify the activity level of $\mathbf{q}_i$. A larger value indicates that query $\mathbf{q}_i$ is more critical within the input sequence $\mathbf{Q}$.

We propose the DVSA mechanism, which allows each key to attend only to the w most important queries determined by the sparsity measure $Sparsity\left(\mathbf{q}_i, \mathbf{K}\right)$. The parameter w is defined as $w = c \cdot \ln L_Q$, with $w = c \cdot \ln L_Q$ being a constant sampling factor. In the new attention formulation, the original sequence $\mathbf{Q}$ is replaced with $\mathbf{Q}_{\mathrm{Top}-w}$, which contains only the top w most significant queries. Consequently, the updated sparse attention mechanism can be given by:

$$Attention\left(\mathbf{Q}, \mathbf{K}, \mathbf{V}\right) = \mathrm{Softmax}\left(\frac{\mathbf{Q}_{\mathrm{Top}-w}\mathbf{K}^{\top}}{\sqrt{d}}\right)\mathbf{V}, \tag{5}$$

This design reduces the time complexity of DVSA to $O\left(L_K \ln L_Q\right)$.

3.4.2 Savitzky-Golay (SG) Filter

In the collection of cloud computing resources, noise is commonly observed, primarily due to the combined influence of multiple factors, with the main ones being as follows. First, the dynamic fluctuations in resource usage make it difficult to maintain stable data. Second, limitations in the precision of collection tools can lead to measurement errors. Lastly, network transmission delays and errors increase the randomness of data collection. These primary factors collectively contribute to the widespread presence of noise in cloud computing resource data collection.

The SG filter can reduce noise in time series data by applying a least-squares polynomial smoothing technique [20,21]. Therefore, the primary function of the SG filter is to preprocess the data, denoising it to enhance the quality and feature representation of the input data.

Assume that the observed value of a specific cloud computing resource at time t is denoted as $\mathcal{X}^t$. Define a subsequence of $\mathcal{X}$ with a window size of $2m + 1$ as $Q_i = \left(\mathcal{X}_{i-m}, \ldots, \mathcal{X}_i, \ldots, \mathcal{X}_{i+m}\right)$. The SG filter utilizes the following polynomial for fitting.

$$\begin{aligned}\mathcal{P}\left(n\right) &= a_0 + a_1 n + a_2 n^2 + \cdots + a_P n^P \\ &= \sum_{p=0}^{P} a_p n^p \quad n \in \left[i - m, i + m\right],\end{aligned} \tag{6}$$

where a_p denotes the coefficient of the p-th term of the polynomial, and P represents the order of the polynomial $\left(P \geq 1\right)$.

To obtain the optimal polynomial coefficients $a_0, a_1, \ldots, a_P$ for each subsequence Q_i, the objective is to minimize the sum of squared errors as defined by:

$$\sum_{j=-m}^{m} \left(\mathcal{X}_{i+j} - \mathcal{P}\left(i + j\right)\right)^2, \tag{7}$$

where $\mathcal{P}\left(i + j\right)$ represents the polynomial approximation for the data point $\mathcal{X}_{i+j}$.

By substituting the polynomial form of $\mathcal{P}\left(i + j\right)$ from Eq. (6) into this sum of squared errors, the following expression is obtained:

$$\min_{\{a_0, a_1, \ldots, a_P\}} \sum_{j=-m}^{m} \left(\mathcal{X}_{i+j} - \sum_{p=0}^{P} a_p \left(i + j\right)^p\right)^2, \tag{8}$$

After iterating through each time slot in the above process, the input data preprocessed by the SG filter can be obtained.

3.4.3 Type-Aware Mixture of Experts (T-MoE)

Different cloud computing resources exhibit distinct time series characteristics. For example, CPU utilization is highly volatile and sensitive to load fluctuations, with pronounced variations in high-concurrency and multi-tasking environments. In contrast, memory usage is relatively stable. Therefore, accurately predicting various types of cloud computing resources and accommodating their diverse requirements requires adjusting model parameters based on the specific attributes of each resource.

To improve the prediction accuracy of cloud computing resources and adapt to the unique characteristics of different resources, we introduce the T-MOE model [22]. The T-MOE model is particularly suitable for this scenario due to its dynamic adaptability when handling complex and diverse input features.

Upon obtaining data for multiple cloud computing resources, model parameters can be tailored to the specific characteristics of each resource. For example, for CPU resources, parameters are optimized to account for their volatility, resulting in the construction of the SDVformer-CPU model. Similarly, for memory resources, parameters are adjusted to accommodate their stability and dynamic variations, leading to the SDVformer-MEM model. During the prediction phase, input data first passes through a gating network, which generates activation weights for each expert module. These weights determine not only which expert modules are activated but also their respective contributions to the final prediction, ensuring a high degree of alignment between resource characteristics and model performance.

The T-MOE model effectively addresses the diverse characteristics of cloud computing resources through its adaptive expert allocation mechanism, achieving efficient and robust resource demand prediction, which aligns with the requirements of cloud computing resource prediction tasks for both accuracy and flexibility.

4 Experiment

4.1 Datasets

To validate the effectiveness of the SDVformer model, we select two datasets: the publicly available Alibaba dataset Cluster-trace-v2018 and the Beijing Jiaotong University (BJTU) self-collected dataset.

Cluster-trace-v2018 contains resource usage information from 4000 machines over 8 days, comprising six files that include metadata and event logs for both machines and containers. We select the 'machine_usage.csv' file to extract CPU and memory usage as prediction metrics. From the 4000 machines, three are randomly chosen (identified as m_1932, m_1989, and m_2025) for analysis. The dataset is divided into six groups: CPU and memory usage for each of m_1932, m_1989, and m_2025.

The BJTU self-collected dataset is obtained from BJTU laboratory servers. This dataset records the CPU and memory usage of three laboratory servers (identified as data_0, data_1, and data_2) during randomly selected periods. Based on server IDs and resource types, the dataset is divided into six groups: CPU and memory usage for each of data_0, data_1, and data_2.

The details of the dataset are summarized in Table 1.

Table 1: Datasets list

Machine ID	Resource type	Dataset type	The samples of datasets
m_1932	CPU	Alibaba public dataset	61,570
m_1932	MEM	Alibaba public dataset	61,570
m_1989	CPU	Alibaba public dataset	62,652
m_1989	MEM	Alibaba public dataset	62,652
m_2025	CPU	Alibaba public dataset	61,898
m_2025	MEM	Alibaba public dataset	61,898
data_0	CPU	BJTU self-collected dataset	14,999
data_0	MEM	BJTU self-collected dataset	14,999
data_1	CPU	BJTU self-collected dataset	49,116
data_1	MEM	BJTU self-collected dataset	49,116
data_2	CPU	BJTU self-collected dataset	179,568
data_2	MEM	BJTU self-collected dataset	179,568

4.2 Experimental Details

We allocate 70% of the time series data in the dataset as the training set, 10% as the validation set, and the remaining 20% as the testing set. Moreover, we select multiple methods to comprehensively compare the performance of SDVformer with other approaches. Specifically, we include traditional time-series forecasting models such as ARIMA, LSTM, and Informer. Furthermore, recent advanced models including Autoformer and FEDformer are also incorporated. A detailed list of models is provided in Table 2.

Table 2: Methods list

Methods	Explanation
ARIMA	AutoRegressive integrated moving average
RNN	Recurrent neural network
LSTM	Long short-term memory
Bi-LSTM	Bidirectional long short-term memory
Informer	A classical representative of Transformer-based variants
Autoformer [23]	A typical Transformer variant with trend-seasonal decomposition.
FEDformer [24]	A typical Transformer variant with frequency-aware decomposition.
SDVformer	Proposed

We use Mean Square Error (MSE) as the indicator to evaluate the performance of the experimental method. The MSE is define as:

$$MSE = \frac{1}{m}\sum_{t=1}^{m}\left(y_t - \hat{y}_t\right)^2, \tag{9}$$

where m denotes the number of samples, $\hat{y}_t$ represents the average of the ground truth values, and y_t refers to the predicted value at time slot t.

Table 3 presents a selection of key model parameters used in the experiments.

Table 3: Key model parameters

Parameter	Value	Description
seq_len	96	Length of historical input time window
label_len	48	Length of decoder's known tokens
enc_in	2	Input dimension of the encoder
dec_in	1	Input dimension of the decoder
c_out	1	Output dimension
e_layers	1	Num of encoder layers
d_layers	1	Num of decoder layers
d_model	512	Dimension of model
sg_len	5	SG Filter window length
sg_po	2	SG Filter polynomial order

4.3 *Experimental Results and Discussion*

4.3.1 Comparison of the Prediction Performance

To evaluate the predictive performance of SDVformer, we test the traditional time-series forecasting models, along with Informer, Autoformer, FEDformer, and SDVformer, on six Alibaba public datasets. The evaluation metrics of the experimental results are presented in Table 4.

Table 4: Performance comparison of traditional models, informer, autoformer, FEDformer, and SDVformer on the Alibaba public dataset (MSE Evaluation Metric, where the smaller the value of the elements in the table, the better the corresponding method)

Datasets\Methods	ARIMA	RNN	LSTM	Bi-LSTM	Informer	Autoformer	FEDformer	SDVformer
m_1932(CPU)	195.32	44.78	43.99	43.96	62.91	54.20	49.84	**1.75**
m_1989(CPU)	178.58	57.19	54.80	55.01	69.18	62.28	59.07	**1.89**
m_2025(CPU)	224.99	63.25	62.82	63.06	62.09	73.35	72.14	**2.42**
m_1932(MEM)	29.21	2.53	2.39	2.41	2.46	3.70	3.48	**0.12**
m_1989(MEM)	64.98	2.66	2.32	2.35	2.15	4.44	3.02	**0.13**
m_2025(MEM)	14.42	2.15	2.12	2.11	2.04	3.32	2.73	**0.094**

As shown in Table 4, the following conclusion can be drawn: SDVformer achieves much higher performance than baseline methods. Compared to traditional models and Informer, the MSE value of SDVformer decreases by approximately 80%, with its MSE values consistently below 10.

4.3.2 Empirical Validation of DVSA and SG Filter in Performance Enhancement

In Section 3.4, we propose two key algorithms of SDVformer: DVSA and the SG filter. To verify the effectiveness of these algorithms in enhancing model performance, we perform experiments.

(1) Empirical Validation of DVSA in Performance Enhancement

In Informer, the attention mechanism algorithm used is ProbSparse Self-Attention. We keep the other components of the Informer architecture unchanged, only replacing the ProbSparse Self-Attention algorithm with the DVSA algorithm. The resulting model is referred to as DVformer.

We test the traditional time-series forecasting models, along with Informer, Autoformer, FEDformer, and DVformer, on six Alibaba public datasets. The evaluation metrics of the experimental results are presented in Table 5.

Table 5: Performance comparison of traditional models, informer, autoformer, FEDformer, and DVformer on the Alibaba public dataset (MSE Evaluation Metric, where the smaller the value of the elements in the table, the better the corresponding method)

Datasets\Methods	ARIMA	RNN	LSTM	Bi-LSTM	Informer	Autoformer	FEDformer	DVformer
m_1932(CPU)	195.32	44.78	43.99	43.96	62.91	54.20	49.84	**42.24**
m_1989(CPU)	178.58	57.19	54.80	55.01	69.18	62.28	59.07	**53.38**
m_2025(CPU)	224.99	63.25	62.82	63.06	62.09	73.35	72.14	**61.62**
m_1932(MEM)	29.21	2.53	2.39	2.41	2.46	3.70	3.48	**2.44**
m_1989(MEM)	64.98	2.66	2.32	2.35	2.15	4.44	3.02	**2.14**
m_2025(MEM)	14.42	2.15	2.12	2.11	2.04	3.32	2.73	**2.03**

As shown in Table 5, we can conclude that: Compared to Informer's ProbSparse Self-Attention, DVSA demonstrates superior performance. Firstly, the MSE of Informer is not always optimal when compared with traditional models and even exhibits the worst overall performance in the CPU dataset. Secondly, by replacing the Informer's self-attention mechanism with DVSA (i.e., DVformer), the MSE values are reduced, and the model outperforms traditional models in most scenarios.

(2) Empirical Validation of SG Filter in Performance Enhancement

In addition to the SG filter, we also select other filters, including the Moving Average Filter and the Gaussian Filter. These three filters were incorporated into the Informer model, and for convenience, we refer to the resulting models as MAIformer, GFIformer, and SGIformer, respectively.

We test the traditional time-series forecasting models, as well as Informer, Autoformer, FEDformer, MAIformer, GFIformer, and SGIformer, on six Alibaba public datasets. The evaluation metrics of the experimental results are presented in Table 6.

Table 6: Performance comparison of traditional models, informer, autoformer, FEDformer, MAIformer, GFIformer, and SGIformer on the Alibaba public dataset (MSE Evaluation Metric, where the smaller the value of the elements in the table, the better the corresponding method)

Datasets\ Methods	ARIMA	RNN	LSTM	Bi-LSTM	Informer	Autoformer	FEDformer	MAIformer	GFIformer	SGIformer
m_1932(CPU)	195.32	44.78	43.99	43.96	62.91	54.20	49.84	11.00	5.10	**1.97**
m_1989(CPU)	178.58	57.19	54.80	55.01	69.18	62.28	59.07	11.09	5.30	**2.01**
m_2025(CPU)	224.99	63.25	62.82	63.06	62.09	73.35	72.14	12.61	6.30	**2.58**
m_1932(MEM)	29.21	2.53	2.39	2.41	2.46	3.70	3.48	0.83	0.39	**0.13**
m_1989(MEM)	64.98	2.66	2.32	2.35	2.15	4.44	3.02	1.39	0.56	**0.16**
m_2025(MEM)	14.42	2.15	2.12	2.11	2.04	3.32	2.73	0.65	0.30	**0.10**

As shown in Table 6, we can conclude that: Combining filters with Informer significantly improves performance, with the SG filter achieving the best results. Specifically, MAlformer, GFlformer, and SGlformer extend Informer by incorporating Moving Average filters, Gaussian filters, and SG filters, respectively. Experimental results demonstrate that the MSE values of these three models are significantly lower compared to the original Informer.

4.3.3 Generalization Experiment

To validate the generalization capability of SDVformer in task resource demand prediction, we conduct further experiments on six BJTU self-collected datasets. The evaluation metrics of the experimental results are presented in Table 7.

Table 7: Performance comparison of methods on the BJTU self-collected dataset (MSE Evaluation Metric, where the smaller the value of the elements in the table, the better the corresponding method)

Methods	ARIMA	RNN	LSTM	Bi-LSTM	Informer	Autoformer	FEDformer	SDVformer
data_0(CPU)	490.25	55.28	58.28	46.71	32.35	80.11	74.20	**2.89**
data_1(CPU)	17,772.60	324.11	143.14	100.68	94.21	78.06	53.58	**5.29**
data_2(CPU)	1561.60	89.04	53.60	40.04	38.25	33.75	35.59	**2.11**
data_0(MEM)	0.038	0.015	0.018	0.020	0.013	0.029	0.016	**0.0011**
data_1(MEM)	24.30	0.34	0.28	0.22	0.073	0.19	0.036	**0.011**
data_2(MEM)	20.46	0.19	0.18	0.21	0.45	0.22	0.039	**0.11**

As shown in Table 7, it can be concluded that SDVformer remains the best-performing model, demonstrating strong generalization capability in cloud computing task resource demand prediction and adaptability to different datasets.

Figs. 2–5 compare the predictive performance of Informer and SDVformer across six Alibaba public datasets. It is evident that SDVformer achieves a significantly better fit between the predicted and true values.

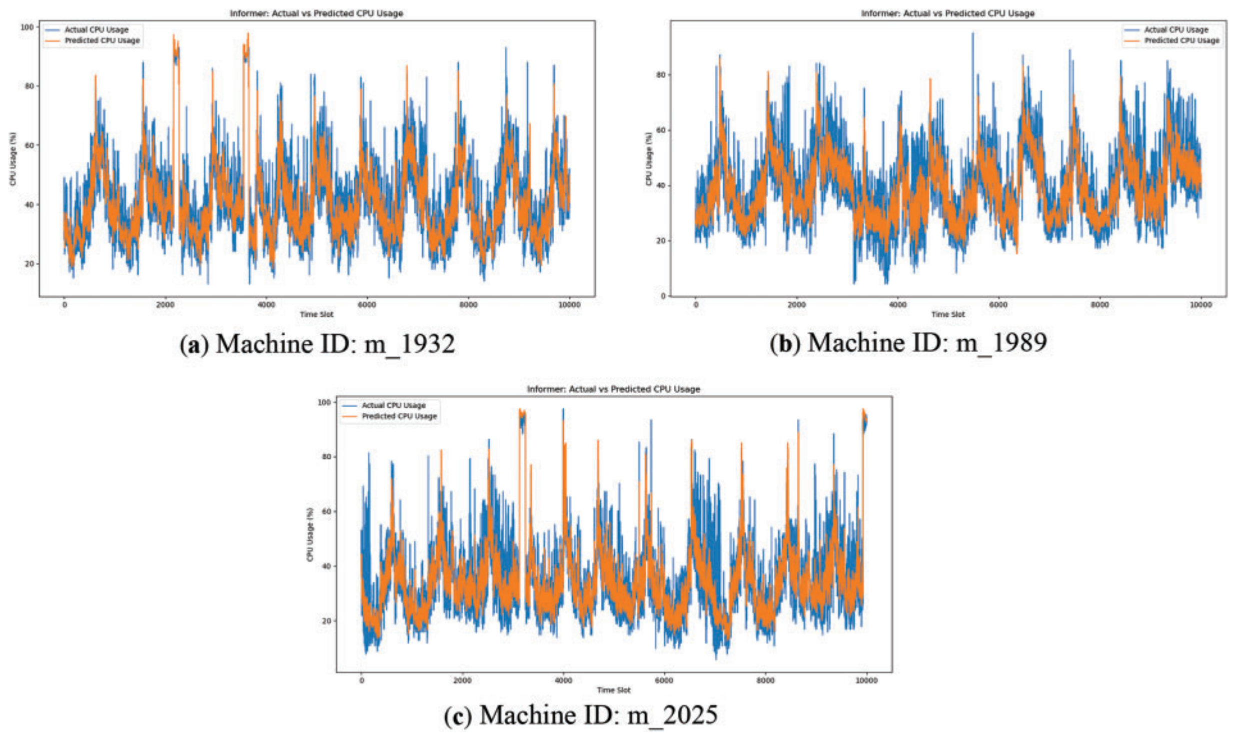
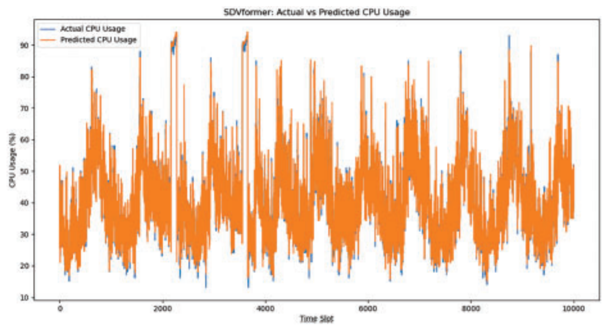
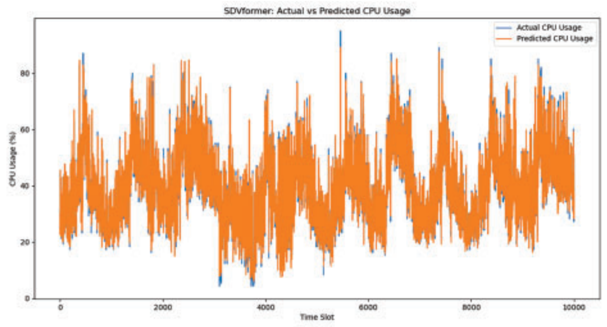


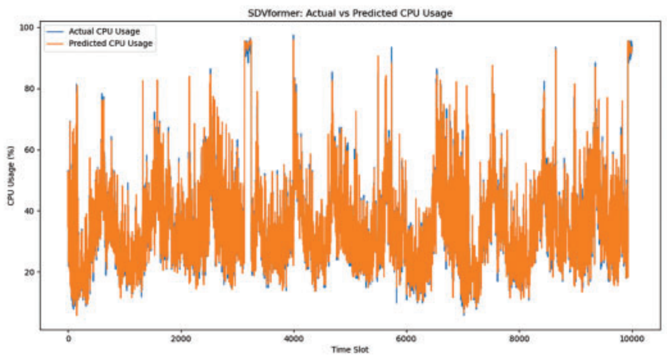
Figure 2: Informer prediction on Alibaba public datasets (CPU)



(a) Machine ID: m_1932

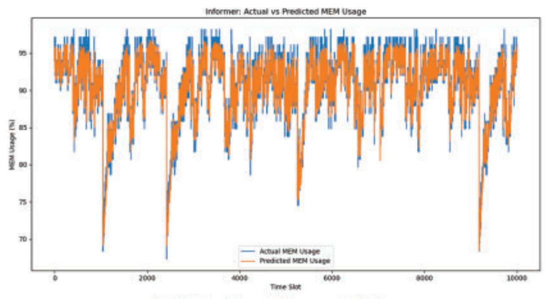


(b) Machine ID: m_1989

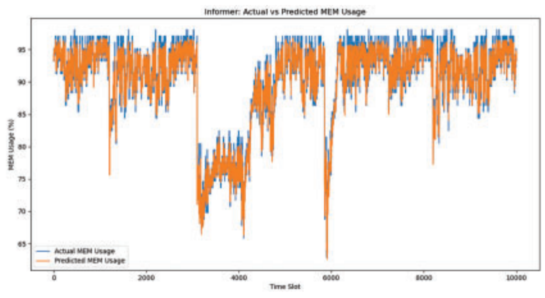


(c) Machine ID: m_2025

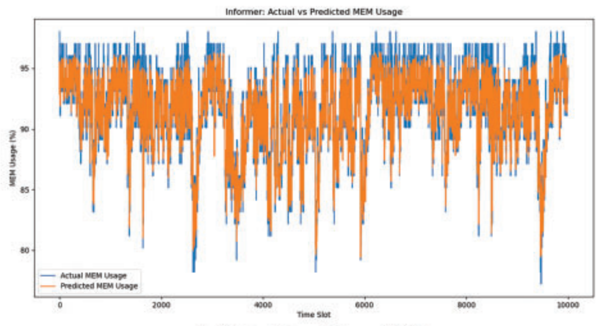
Figure 3: SDVformer prediction on Alibaba public datasets (CPU)



(a) Machine ID: m_1932

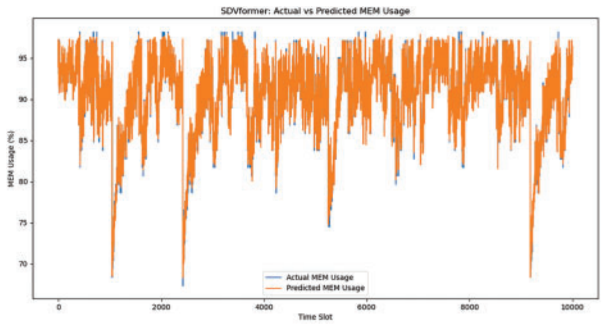


(b) Machine ID: m_1989

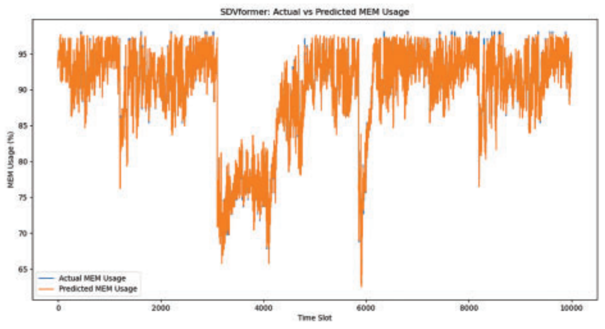


(c) Machine ID: m_2025

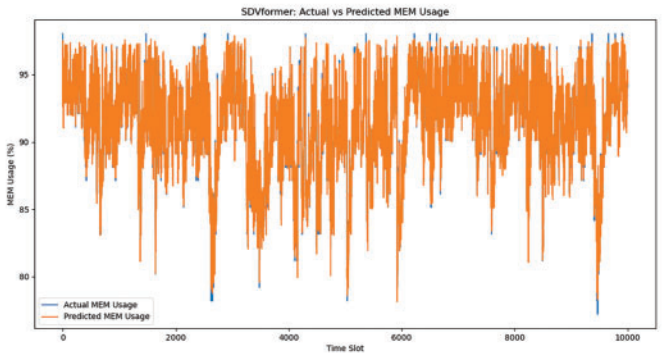
Figure 4: Informer prediction on Alibaba public datasets (MEM)



(**a**) Machine ID: m_1932



(**b**) Machine ID: m_1989



(**c**) Machine ID: m_2025

Figure 5: SDVformer prediction on Alibaba public datasets (MEM)

5 Conclusions and Future Work

In this paper, we proposed SDVformer, an enhanced time series forecasting model based on the Informer architecture to predict resource utilization in cloud systems. SDVformer incorporated a DVSA mechanism, an SG filter, and a T-MOE framework. The new self-attention mechanism used variance formulas to identify key queries **Q**, capturing critical features while reducing computational costs. The SG filter reduces noise and improves data preprocessing robustness. The T-MOE framework enhanced adaptability by customizing expert modules for resource-specific characteristics, improving prediction accuracy. Experimental results on Alibaba and BJTU self-collected datasets demonstrated that SDVformer outperformed models such as RNN and Informer, achieving a reduction of approximately 80% in MSE. The current time and space complexity of SDVformer is comparable to that of Informer, future work will focus on further reducing its computational complexity to enhance efficiency in large-scale cloud computing scenarios.

Acknowledgement: Not applicable.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: Shui Liu prepared the figures. Shui Liu and Ke Xiong designed the model, Shui Liu wrote the paper, Shui Liu, Ke Xiong and Yeshen Li revised the paper. Ke Xiong, Zhifei Zhang, Yu Zhang, and Pingyi Fan supervised the project. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The public data that support the findings of this study are openly available in Alibaba dataset Cluster-trace-v2018 at https://github.com/alibaba/clusterdata/tree/v2018/cluster-trace-v2018 (accessed on 01 June 2025). The BJTU self-collected data that support the findings of this study are available from the Corresponding Author, K. X., upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Sabyasachi AS, Muppala JK. Cost-effective and energy-aware resource allocation in cloud data centers. Electronics. 2022;11(21):3639. doi:10.3390/electronics11213639.
2. Jayanetti A, Halgamuge S, Buyya R. Multi-agent deep reinforcement learning framework for renewable energy-aware workflow scheduling on distributed cloud data centers. IEEE Trans Parallel Distrib Syst. 2024;35(4):604–15. doi:10.1109/tpds.2024.3360448.
3. Xia Y, Zhou M, Luo X, Zhu Q, Li J, Huang Y. Stochastic modeling and quality evaluation of infrastructure-as-a-service clouds. IEEE Trans Automat Sci Eng. 2015;12(1):162–70. doi:10.1109/tase.2013.2276477.
4. Yuan H, Bi J, Zhou M. Profit-sensitive spatial scheduling of multi-application tasks in distributed green clouds. IEEE Trans Autom Sci Eng. 2020;17(3):1097–106. doi:10.1109/TASE.2019.2909866.
5. Kashyap RL. Optimal choice of AR and MA parts in autoregressive moving average models. IEEE Trans Pattern Anal Mach Intell. 1982;2(2):99–104. doi:10.1109/tpami.1982.4767213.
6. Zhang J, Man KF. Time series prediction using RNN in multi-dimension embedding phase space. In: Proceedings of the SMC'98 Conference Proceedings; 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218); 1998 Oct 14; San Diego, CA, USA. doi:10.1109/icsmc.1998.728168.
7. Gers FA, Schmidhuber J, Cummins F. Learning to forget: continual prediction with LSTM. Neural Comput. 2000;12(10):2451–71. doi:10.1162/089976600300015015.
8. Siami-Namini S, Tavakoli N, Namin AS. The performance of LSTM and BiLSTM in forecasting time series. In: Proceedings of the 2019 IEEE International Conference on Big Data (Big Data); 2019 Dec 9–12; Los Angeles, CA, USA. doi:10.1109/bigdata47090.2019.9005997.
9. Guo J, Chang Z, Wang S, Ding H, Feng Y, Mao L, et al. Who limits the resource efficiency of my datacenter: an analysis of Alibaba datacenter traces. In: Proceedings of the International Symposium on Quality of Service; 2019 Jun 24; New York, NY, USA.
10. Zhang GP. Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing. 2003;50:159–75. doi:10.1016/s0925-2312(01)00702-0.
11. Calheiros RN, Masoumi E, Ranjan R, Buyya R. Workload prediction using ARIMA model and its impact on cloud applications' QoS. IEEE Trans Cloud Comput. 2015;3(4):449–58. doi:10.1109/tcc.2014.2350475.
12. Qiu X, Zhang L, Nagaratnam Suganthan P, Amaratunga GAJ. Oblique random forest ensemble via Least Square Estimation for time series forecasting. Inf Sci. 2017;420(2):249–62. doi:10.1016/j.ins.2017.08.060.
13. Mallick S, Hains G, Deme CS. A resource prediction model for virtualization servers. In: Proceedings of the 2012 International Conference on High Performance Computing & Simulation (HPCS); 2012 Jul 2–6; Madrid, Spain. doi:10.1109/hpcsim.2012.6266990.
14. Hu R, Jiang J, Liu G, Wang L. CPU load prediction using support vector regression and Kalman smoother for cloud. In: Proceedings of the 2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops; 2013 Jul 8–11; Philadelphia, PA, USA. doi:10.1109/icdcsw.2013.60.
15. Duggan M, Mason K, Duggan J, Howley E, Barrett E. Predicting host CPU utilization in cloud computing using recurrent neural networks. In: Proceedings of the 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST); 2017 Dec 11–14; Cambridge, UK. doi:10.23919/icitst.2017.8356348.
16. Karim ME, Maswood MMS, Das S, Alharbi AG. BHyPreC: a novel Bi-LSTM based hybrid recurrent neural network model to predict the CPU workload of cloud virtual machine. IEEE Access. 2021;9:131476–95. doi:10.1109/access.2021.3113714.

17. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017); 2017 Dec 4–9; Long Beach, CA, USA.
18. Zhou H, Zhang S, Peng J, Zhang S, Li J, Xiong H, et al. Informer: beyond efficient transformer for long sequence time-series forecasting. AAAI [Internet]. 2021 May 18 [cited 2025 Jun 1]. Available from: https://ojs.aaai.org/index.php/AAAI/article/view/17325.
19. Zhao F, Lin W, Lin S, Tang S, Li K. MSCNet: multi-scale network with convolutions for long-term cloud workload prediction. IEEE Trans Serv Comput. 2025;18(2):969–82. doi:10.1109/TSC.2025.3536313.
20. Savitzky A, Golay MJ. Smoothing and differentiation of data by simplified least squares procedures. Anal Chem. 1964;36(8):1627–39. doi:10.1021/ac60214a047.
21. Bi J, Ma H, Yuan H, Buyya R, Yang J, Zhang J, et al. Multivariate resource usage prediction with frequency-enhanced and attention-assisted transformer in cloud computing systems. IEEE Internet Things J. 2024;11(15):26419–29. doi:10.1109/jiot.2024.3395610.
22. Jordan MI, Jacobs RA. Hierarchical mixtures of experts and the EM algorithm. Neural Comput. 1994;6(2):181–214. doi:10.1162/neco.1994.6.2.181.
23. Wu H, Xu J, Wang J, Long M. Autoformer: decomposition transformers with auto-correlation for long-term series forecasting. Adv Neural Inf Process Syst. 2021;34:22419–30.
24. Zhou T, Ma Z, Wen Q, Wang X, Sun L, Jin R. Fedformer: frequency enhanced decomposed transformer for long-term series forecasting. In: Proceedings of the International Conference Machine Learning (ICML); 2022 Jul 17–23; Baltimore, MD, USA.