Cloud Workload Prediction Based on Bayesian-Optimized Autoformer

Biying Zhang, Yuling Huang, Zuoqiang Du*, Zhimin Qiu
School of Computer and Information Engineering, Harbin University of Commerce, Harbin 150028, China

***Abstract*—Accurate workload forecasting plays a pivotal role in the management of cloud computing resources, enabling significant enhancement in the performance of the cloud platform and effective prevention of resource wastage. However, the complexity, variability, and strong time dependencies of cloud workloads make prediction difficult. To address the challenge of enhancing accuracy in contemporary cloud workload prediction, this paper employs empirical and quantitative research methods, introducing a cloud workload prediction method based on Bayesian-optimized Autoformer, termed BO-Autoformer. Initially, the cloud workload data were divided according to the time-sliding window to construct a continuous feature sequence, which was used as the input of the model to construct the Autoformer prediction model. Subsequently, to further enhance the model's performance, the Bayesian optimization method was employed to identify the optimal combination of hyperparameters, resulting in the development of the Bayesian optimization-based Autoformer cloud workload prediction model. Finally, experiments were conducted on a real Google dataset to evaluate the model's effectiveness. The findings reveal that, compared to alternative models, the proposed prediction model demonstrates superior performance on the cloud workload dataset, and can effectively improve the prediction accuracy of the cloud workload.**

***Keywords*—Cloud computing; deep learning; workload prediction; Autoformer; Bayesian optimization**

I. INTRODUCTION

Cloud computing plays a crucial role in promoting public availability and openness of computing resources [1], yet the low utilization of these resources remains a persistent challenge in cloud computing resource management. With the continuous expansion of cloud computing infrastructure and the rapid increase in the number of users, energy consumption in cloud data centers has emerged as a significant issue [2]. The utilization of physical hosts is a key factor that substantially influences the energy consumption of the entire cloud computing system. Resource over-allocation or under-utilization markedly escalates energy costs within cloud data centers. Studies indicate that the current utilization rate of various cloud computing resources generally falls below 50%, leading to a substantial portion of these resources remaining idle [3]. This inefficiency in resource usage not only results in considerable wastage of societal resources but also underscores the urgent need for implementing effective strategies to enhance the efficiency of cloud computing resource utilization. Therefore, effective measures must be taken to lower energy consumption costs, minimize resource waste, and foster the sustainable development of cloud computing.

An effective approach to enhance the utilization of cloud computing resources involves accurate prediction of the resource workload. Through an analysis of historical data pertaining to the usage of cloud computing resources, it is possible to uncover the underlying patterns of load fluctuations, thereby forecasting the workload of cloud computing resources in the upcoming period [4]. By leveraging these predictive insights, cloud service providers can proactively adjust resource allocation to meet the diverse needs of users while optimizing resource utilization efficiency. However, the pursuit of predictive accuracy faces a series of challenges. First, workload fluctuations are influenced by numerous factors, such as the unpredictability of user behavior, sudden business demands, and the dynamic allocation of system resources. These factors result in workload patterns that are difficult to accurately capture with simple models. Second, Cloud workload prediction relies on extensive historical data to train predictive models, yet ensuring the integrity, accuracy, and consistency of this data is often challenging. Issues such as missing values, outliers, and noise are prevalent, which can interfere with the model training process and lead to biased prediction results. Consequently, traditional prediction models often struggle to achieve the desired accuracy when addressing the complexity and dynamics of cloud workloads and the instability of data quality. To solve this problem, it is imperative to explore new predictive technologies and methods to enhance the accuracy and reliability of prediction models.

The Autoformer, a Transformer-based method for time series prediction, was introduced by Wu Haixu and colleagues from Tsinghua University [5]. It incorporates an autocorrelation mechanism to capture temporal dependencies, thereby enhancing its predictive accuracy and efficiency. Among existing time series forecasting methods, Autoformer has garnered widespread attention due to its outstanding performance. However, its application in cloud computing workload prediction is hindered by challenges in hyperparameter tuning and performance fluctuations:

- Hyperparameter Adjustment Difficulties: The performance of the Autoformer depends significantly on the appropriate selection and tuning of hyperparameters. However, manual adjustment of hyperparameters becomes exceedingly challenging and time-consuming, particularly in scenarios involving voluminous data and multifaceted tasks.

*Corresponding Author
Fund Project: Science and Research Project of Harbin University of Commerce [grant no. 2019DS032]

- Performance Fluctuation: The performance of the Autoformer can fluctuate considerably across different tasks and datasets due to the complexity of its structure and the diversity of input datasets, thereby constraining its broad applicability in cloud computing workload prediction.

This paper proposes a cloud workload prediction method that optimizes Autoformer through Bayesian optimization technology, which aims to solve the above problems. The main contribution of this work can be summarized as follows:

- The Autoformer model is first applied to the field of cloud computing workload prediction, which significantly improves the accuracy and efficiency of prediction by exploiting its unique self-attention mechanism and long sequence processing ability.
- Bayesian optimization technique is used to optimize the combination of hyperparameters of Autoformer. Compared with traditional grid search or random search methods, Bayesian optimization techniques can more effectively explore the hyperparameter space and find the combination of hyperparameters that optimizes the model performance.

The application of the Bayesian optimization technique enables the Autoformer model to automatically tune hyperparameters in the workload prediction task of cloud computing. Through automatic tuning, not only the tedious and time-consuming manual parameter adjustment is eliminated, but also the computational cost in the tuning process is greatly reduced. With the help of Bayesian optimization, Autoformer can quickly find the best combination of hyperparameters for a specific task, thereby improving the accuracy of prediction. Furthermore, Bayesian optimization enhances the stability of the Autoformer model across various datasets and tasks, thereby mitigating performance fluctuations and bolstering the reliability of cloud computing systems. Consequently, the optimized Autoformer model can deliver more precise and dependable prediction outcomes across diverse workload scenarios. For cloud computing service providers, this technological improvement means that they can allocate resources more efficiently, and avoid resource waste or over-provisioning, thereby reducing costs and improving service quality. At the same time, the system operation and maintenance personnel can also understand the load of the system in advance with the help of the BO-Autoformer model, so that they can make timely responses and adjustments, reducing the risk of system downtime or performance degradation. In addition, the combination of Bayesian optimization and the Autoformer model also provides new ideas and methods for research in the field of cloud computing and machine learning and encourages scholars and engineers in related fields to conduct more in-depth research and exploration.

The structure of this paper is as follows: Section II provides a review of relevant literature. Section III elucidates the foundational principles of the Autoformer model. Section IV examines the application of the Autoformer model to workload predictions, with an emphasis on Bayesian optimization. Section V discusses the results derived from experimental evaluations. Section VI concludes the paper with a summary of the findings and contributions.

II. Related Works

In recent years, native and overseas scholars have dedicated efforts to enhancing the accuracy and reliability of workload forecasting for cloud computing resources. The methodologies employed in these studies can be broadly categorized into three primary groups: traditional regression techniques, machine learning approaches, and deep learning strategies. These methodologies represent the evolving landscape of research in the realm of cloud computing resource load prediction, reflecting a progression from conventional statistical methods to more sophisticated artificial intelligence models.

A. Traditional Regression Techniques

Traditional regression techniques encompass a diverse array of methodologies, including Autoregressive[6](AR), Moving average[7](MA), Autoregressive moving average[8](ARMA), Differential autoregressive moving average method[9](ARIMA), Linear regression[10](LR) and Exponential smoothing[11](ES), etc. Predominantly grounded in the presumption of linear interrelations, these methods frequently fall short of capturing the non-linear dynamics of workload variations. A significant reliance on the principle of stationarity renders them less effective in managing the non-stationary nature of cloud resource loads. Challenges such as the complexity of parameter selection, susceptibility to outliers, and the difficulty in addressing seasonality and trends further underscore the limitations of traditional regression techniques in the context of cloud computing resource load forecasting.

B. Machine Learning Approaches

Machine learning approaches primarily comprise Markov models [12], Bayesian models [13, 14], Support vector regression (SVR) models[15], and traditional Artificial neural networks [16](ANN). In comparison to deep learning techniques, machine learning methods are somewhat constrained in their capacity to navigate complex non-linear relationships, exhibit limitations in effective feature extraction, and demonstrate inferior generalization performance in predictive models. These methods commonly call for a substantial amount of training data to obtain good prediction performance, and the data mostly rely on heuristic algorithms. As such, precise predictions require workload data that exhibits clear regularities or patterns, highlighting the dependency of machine learning methods on the characteristic structure of the data.

C. Deep Learning Strategies

Deep learning strategies encompass a variety of models, including Recurrent neural network [17](RNN), Long short-term memory network (LSTM), Gated recurrent unit[18](GRU), Convolutional neural network [19](CNN), and Deep belief network [20](DBN), etc. To overcome the challenges of gradient vanishing and exploding encountered during the training of recurrent neural networks, researchers proposed the Long Short-Term Memory network (LSTM). By introducing a gating mechanism, LSTM can better manage and utilize gradient information, thereby enabling the learning of more intricate temporal patterns and prolonged dependencies.

Guo et al. [21] proposed an enhanced model, N-LSTM, which is an extension of LSTM, specifically designed to address the challenge of virtual machine workload prediction. This model integrates historical VM workload data with request intervals across various VM categories, using the resulting dataset as input for training, thereby enhancing its ability to accurately forecast future VM workload. In an effort to address the limitations and challenges present in current research, Xu et al.[22] introduced a deep neural network method, referred to as esDNN, based on efficient supervised learning for cloud workload prediction. Empirical results indicate that esDNN is capable of providing precise and efficient predictions for cloud workload.

Following the advent of the Transformer [23] model in the realms of natural language processing and computer vision, a multitude of Transformer-based models have been adapted for the prediction of time series data [24–26]. The Transformer architecture capitalizes on the distinct capabilities of the attention mechanism to effectively discern global dependencies within sequences, thus offering enhanced performance in addressing long-term sequence prediction challenges. However, the direct application of the self-attention mechanism in these models presents notable challenges in accurately discerning time dependencies within complex time series patterns. Furthermore, the intrinsic quadratic complexity associated with the self-attention mechanism imposes limitations on the model's sparsity requirements, thereby influencing the efficiency of information utilization.

In response to these problems, the industry proposed the Autoformer model, which emerged as a significant advancement in time series prediction, extending the horizon of predictability. Empirical research has validated the capability of Autoformer to elevate both the accuracy and efficiency of predictions. To date, the Autoformer has been successfully implemented in various domains, including temperature forecasting, water level prediction, traffic flow forecasting, and power load forecasting. Theoretically, it can also be applied to cloud workload data with time series characteristics to make up for deficiencies in actual cloud load data prediction work. However, there is a dearth of literature demonstrating its application within the cloud computing workload prediction sphere. At the same time, traditionally, tuning the hyperparameter of Autoformer poses a significant challenge, particularly in scenarios involving large-scale datasets and complex tasks. Bayesian optimization, as an intelligent approach grounded in Bayes' theorem, offers potential solutions to a wide range of challenges. By constructing and dynamically refining the probability model of the objective function, this method adeptly incorporates historical evaluation data while efficiently guiding the search process toward a swift convergence to the optimal solution [27-28]. Bayesian optimization has been successfully applied to hyperparameter tuning of multiple predictive models, significantly improving model accuracy [29-32]. Consequently, the introduction of a Bayesian-optimized Autoformer model into cloud computing workload prediction tasks may bring new breakthroughs and improvements in this field of research.

III. AUTOFORMER MODEL PRINCIPLE

The challenge of predicting cloud workload shares similarities with time series prediction, both necessitating the input of a historical time window of length *I* to forecast a future time window of length *O*. In the context of cloud workload prediction, long-term forecasting assumes particular significance as cloud service providers need to proactively plan resource allocation. Addressing the exigency of long-term prediction, this paper introduced the Autoformer model, depicted in Fig. 1. Leveraging a decomposition architecture and autocorrelation mechanism, Autoformer adeptly handles intricate temporal patterns, discerns periodicity, and captures dependencies within time series data. This approach enhances the accuracy and efficiency of cloud load forecasting, enabling cloud service providers to adapt adeptly to forthcoming demand fluctuations.

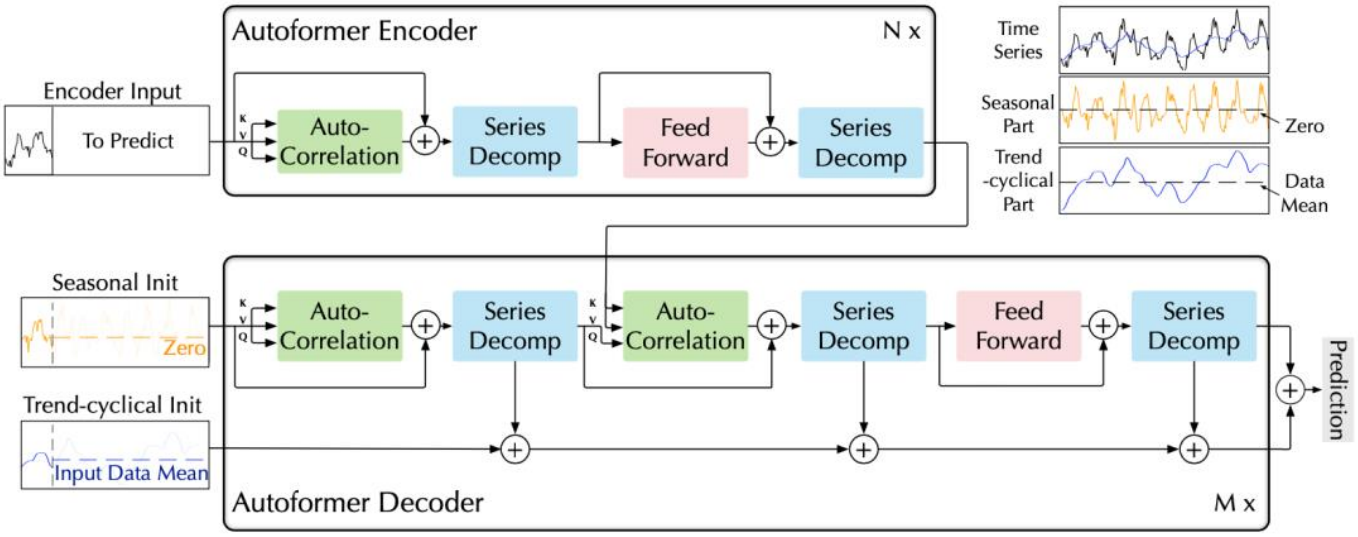

Fig. 1. Cloud workload prediction architecture based on Autoformer[5].

The architecture for cloud workload prediction, utilizing Autoformer, comprises two distinct sections: the encoder and the decoder. The encoder primarily focuses on the load cycle term, taking as input the preceding *I*-time steps. Employing the autocorrelation mechanism, the initial input cloud workload time series undergoes preliminary sequence decomposition, resulting in load trend and load cycle components. These components, reflective of long-term trend and periodicity respectively, are then transmitted to the decoder, thereby dividing the input of the decoder into two parts: load trend item and load cycle item. The load cycle item forwarded to the decoder undergoes further sequence decomposition, facilitated by the autocorrelation mechanism. Meanwhile, the load trend item extracts trend information via accumulation operations. The decoder comprises multiple decoding layers, each acquiring a comprehensive segment of load timing data instead of discrete data points. This enables a stepwise sequence decomposition process, augmenting the reliability and precision of load prediction outcomes.

A. Deep Decomposition Architecture

The Autoformer model integrates sequence decomposition as a fundamental component of its deep decomposition architecture, which is embedded in both the encoder and decoder modules. Throughout the prediction process, the model iteratively optimizes forecast results while performing sequence decomposition. This iterative procedure entails meticulous adjustments to comprehensively capture diverse trends and periodicities in the data, effectively isolating essential features.

The concept of sequence decomposition is primarily derived from conventional time series prediction algorithms, such as Arima and Fbprophet. In essence, these traditional algorithms approach the decomposition of time series from a statistical standpoint and give different physical meanings to the decomposed sub-terms, such as trend term, seasonal term, residual term, etc. Therefore, the general form of traditional time series decomposition is shown in Eq. (1):

$$X(t)=T(t)+S(t)+R(t) \tag{1}$$

In Eq. (1), $X(t)$ represents the time series that is subject to decomposition, whereas $T(t)$, $S(t)$, and $R(t)$ corresponds to the trend term, seasonal term, and residual term, respectively, arising from the decomposition process.

Within the context of cloud computing workload prediction, the computation of sequence decomposition within the Autoformer model is detailed in Eq. (2) to Eq. (3):

$$X_t=AvgPool(Padding(X)) \tag{2}$$

$$X_s=X-X_t \tag{3}$$

In the aforementioned formula, X denotes the time series of cloud load to be decomposed, X_t represents the load trend component, whereas X_s signifies the load cycle component. Eq. (2) elucidates the specific algorithm for extracting the trend item X_t: To maintain the temporal span of the load time series unaltered, the fill operation is first applied to X and then AvgPool processing is performed. Eq. (3) clarifies that Autoformer adopts the simplest additive model and solely decomposes two sub-terms: trend term and periodic term. In other words, the input load time series X is subtracted from the obtained load trend term X_t, and the load cycle term X_s is derived using Eq. (2).

1) Encoder: In the task of cloud computing workload prediction, the encoder primarily targets the periodic component of the cloud load time series. Through a meticulous multi-layer sequence decomposition module, it systematically eradicates the trend term from the load time series, ultimately extracting the periodic term. This extracted periodic term serves as valuable load period information, guiding the decoder in its prediction of future load. Taking the *l-th* coding layer X_{en}^{l} as an example, assuming there are N coding layers, the calculation of the *l-th* coding layer is as shown in Eq. (4) to Eq. (5):

$$S_{\text{en}}^{l,1},\_ = SD(AC(X_{en}^{l-1}) + X_{en}^{l-1}) \tag{4}$$

$$S_{\text{en}}^{l,2},\_ = SD(FF(S_{en}^{l,1}) + X_{en}^{l,1}) \tag{5}$$

In the above formula, AC stands for AutoCorrelation processing, SD represents SeriesDecomp processing, and FF denotes FeedForward processing. X_{en}^{l-1} signifies the cloud load time series input during the initial encoding stage. $S_{en}^{l,i}$, where $i \in \{1,2\}$ denotes the *i-th* encoded load cycle information within the encoding layer.

2) Decoder: In the task of cloud computing workload prediction, the decoder comprises two distinct components. The first component is responsible for handling the load trend term outputted by the encoder, gradually extracting trend information from the predicted latent variable through an accumulation operation. The second component focuses on the periodic term outputted by the encoder, employing a stacked autocorrelation mechanism for dependency mining and aggregation of similar subprocesses. Considering the *l-th* decoding layer X_{de}^{l} as an example, and assuming a total of M decoding layers, the decoding layer primarily operates on the input cloud load time series and the encoding layer output. The calculation of the *l-th* decoding layer is detailed in Eq. (6) to Eq. (9):

$$S_{de}^{l,1}, T_{de}^{l,1} = SD(AC(X_{de}^{l-1}) + X_{de}^{l-1}) \tag{6}$$

$$S_{de}^{l,2}, T_{de}^{l,2} = SD(AC(S_{de}^{l,1}, X_{en}^{N}) + S_{de}^{l,1}) \tag{7}$$

$$S_{\text{de}}^{l,3}, T_{de}^{l,3} = SD(FF(S_{de}^{l,2}) + S_{de}^{l,2}) \tag{8}$$

$$T_{de}^{l} = T_{de}^{l-1} + W_{l,1} * T_{de}^{l,1} + W_{l,2} * T_{de}^{l,2} + W_{l,3} * T_{de}^{l,3} \tag{9}$$

In the above formula, AC stands for AutoCorrelation processing, SD represents SeriesDecomp processing, and FF denotes FeedForward processing. X_{de}^{l-1} signifies the cloud load time series input during the initial encoding stage. $S_{de}^{l,i}$ and $T_{de}^{l,i}$, where, $i \in \{1,2,3\}$ denotes the *i-th* load cycle information and

load trend information decoded in the decoding layer. $W_{l,i}$, where, $i \in \{1,2,3\}$ denotes the weight of the *i-th* load trend.

Utilizing the aforementioned progressive decomposition architecture, the Autoformer model effectively captures periodicity and trend variations within the load time series by methodically decomposing latent variables during the process of cloud load prediction. By incorporating the autocorrelation mechanism and accumulation method, the model is able to extract prediction outcomes pertaining to both load cycle item and load trend item, subsequently enabling a more precise prediction of cloud load time series. This alternating process of decomposition and optimization of prediction results mutually reinforces each other, offering robust support for enhancing the overall performance of the model.

B. Auto-Correlation Mechanism

In the context of cloud computing workload prediction, the Autoformer model leverages an autocorrelation mechanism to effectively capture periodic patterns within cloud load time series. Specifically, the autocorrelation module computes the autocorrelation coefficient of the load sequence, enabling the discovery of periodic dependencies. Furthermore, it employs time translation techniques to aggregate similar subload sequences, thereby enhancing comprehension of the model and prediction accuracy of cloud load dynamics. This approach significantly improves the ability of the model to capture intricate temporal patterns and make accurate predictions, which is crucial for effective resource management in cloud computing environments.

In the practical computation of autocorrelation, the Autoformer model employs a fast Fourier transform (FFT) technique to efficiently calculate the autocorrelation coefficient. Initially, the input load time series X_t undergoes mapping to *Q*, *K*, and *V*, followed by conversion into the frequency domain. In the frequency domain, the translation similarity can be calculated more conveniently, which helps to improve the computational efficiency. The specific calculation process is shown in Eq. (10) to Eq. (11):

$$S_{XX}(f) = F(X_t)F^*(X_t) = \int_{-\infty}^{\infty} X_t e^{-i2\pi tf}\,dt \overline{\int_{-\infty}^{\infty} X_t e^{-i2\pi tf}\,dt} \quad (10)$$

Eq. (10), X_t represents the load time series that exhibits periodicity; F denotes the Fourier transform (FFT), while F^* represents its conjugate operation; The variable f signifies the frequency, which is multiplied by 2π to obtain the angular frequency; The multiplication of F and F^* with the respective integration results of the load trend terms facilitates the transformation of the time series into the frequency domain.

$$R_{XX}(\tau) = F^{-1}(S_{XX}(f)) = \int_{-\infty}^{\infty} S_{XX}(f) e^{i2\pi f\tau}\,df \quad (11)$$

Eq. (11), $R_{XX}(\tau)$represents the similarity between the sequence X_t and its τ delay $X_{t-\tau}$, this delay similarity can be regarded as the confidence of the unnormalized period estimate, that is, the confidence $R(\tau)$ of the period length τ. F^{-1} denotes the inverse Fourier transform; The variable f represents frequency, multiplied by 2π to obtain the angular frequency result. Subsequently, an inverse Fourier transform is applied to the outcome derived from Eq. (11), leading to the computation of the autocorrelation coefficient. This approach effectively reduces the computational complexity associated with the autocorrelation solution, thereby enhancing the efficiency and practicality of the overall analysis.

IV. Workload Prediction Model Based on Bayesian-Optimized Autoformer

A. Bayesian Optimization Algorithm

In the context of forecasting cloud computing workload based on the Autoformer model, the selection of hyperparameters holds a pivotal role in relation to model evaluation. The training process necessitates meticulous control and adjustment of numerous hyperparameters, ensuring that the model performs at its optimal level. This fine-tuning is crucial for enhancing the predictive accuracy and overall performance of the Autoformer in handling cloud workload prediction tasks.

Bayesian Optimization (BO) stands as a sequential model-based optimization method designed for black-box function optimization tasks. It is employed to optimize unknown objective functions efficiently, aiming to expedite the discovery of globally optimal solutions with fewer function evaluations. As a result, Bayesian optimization finds widespread application in hyperparameter tuning for machine learning models. The core principle of this approach lies in the utilization of the Gaussian Process as a prior model to approximate the unknown objective function. Through iterative evaluations and modeling of the objective function, Bayesian optimization selects the most promising input point for subsequent evaluation, guided by the current confidence of the model. This selection process, known as the Sampling Strategy or Acquisition Function, is crucial in guiding the search toward optimal solutions. Eq. (12) presents the mathematical formalism underlying this calculation.

$$X^* = \arg\max_{X \in A} f(X) \quad (12)$$

In Eq. (12), X^* represents the optimal parameter set, A denotes the possible set, and $f(X)$ serves as the prior distribution model.

In comparison to grid search and random search, the Bayesian optimization algorithm demonstrates the capability of attaining satisfactory optimization outcomes with a significantly reduced number of iterations. This efficiency is particularly advantageous in scenarios where computational resources are limited or where rapid convergence is desired. The pseudocode of the Bayesian optimization algorithm is presented in Table I, providing a concise and structured overview of the algorithm's operational steps.

Firstly, an initial set of candidate solutions is uniformly selected within the entire feasible domain, typically comprising n_0 points. This serves as the starting point for the subsequent optimization process. Subsequently, a loop iteration is initiated, during which one point is added at each iteration until a total of N candidate solutions are obtained. To determine the next point to evaluate, the already-found candidate solutions are leveraged to establish a Gaussian regression model. This model allows us

to estimate the posterior probability of the function value at any given point. Based on this posterior probability, an acquisition function is formulated, and the point corresponding to the maximum value of this function is chosen as the next search point. Once the next search point is identified, its function value is computed and incorporated into the set of candidate solutions. Finally, the algorithm terminates, returning the maximum value among the N candidate solutions as the optimal solution. This process ensures efficient exploration of the search space and convergence towards the globally optimal solution.

TABLE I. PSEUDOCODE OF BAYESIAN OPTIMIZATION ALGORITHM

Algorithm 1: Bayesian Optimization Algorithm

Select n_0 sampling points and compute the corresponding values of f(x)

$n=n_0$

While ($n \leqslant N$) do

  Modify the mean and variance of $p(f(x)|D)$ according to the recent Sampling records $D=\{(x_i,f(x_i)),i=1,\ldots,n\}$

  Determine the acquisition function u(x) using the mean and variance of the conditional probability $p(f(x)|D)$

  Identify the subsequent sampling point $x_{n+1}=\text{argmax } u(x)$ by locating the maximum value of the acquisition function

  Compute the function's output at the subsequent sampling location: $y_n=f(x_{n+1})$

  n=n+1

End

Return: $\text{argmax}(f(x_1),\cdots,f(x_N))$ and the corresponding y

B. Workload Prediction Model Based on BO-Autoformer

To enhance the workload prediction model by optimizing its hyperparameters, the Bayesian (BO) algorithm is introduced for parameter optimization. The process of load prediction using the BO-Autoformer prediction model is depicted in Fig. 2. The BO-Autoformer prediction model utilized comprises four components: preprocessing of data, training of the model, Bayesian optimization, and model predictions. Collectively, these stages form a comprehensive load prediction process. The detailed ideas are outlined as follows:

Step 1: In the workload prediction task, the initial load data necessitates rigorous preprocessing. This involves crucial steps such as data cleansing, handling missing values, normalization, and feature engineering. These processes ensure that the load data is rendered suitable for effective training and prediction.

Step 2: Following data preprocessing, the workload dataset is partitioned into distinct training and testing sets. The training set is then utilized to train the Autoformer model, a time series prediction model grounded in stochastic process theory. The Autoformer boasts a deep decomposition architecture and an autocorrelation mechanism, enabling it to efficiently leverage the periodicity and delay information inherent in the sequence.

Step 3: Concurrently, during the model training phase, the Bayesian optimization algorithm is employed to fine-tune the hyperparameters of the Autoformer model. Bayesian optimization establishes a Gaussian regression model within the parameter space, estimating the posterior probability distribution of the objective function. Based on this distribution, it selects the most promising parameter point for the next iteration. This approach significantly enhances the efficiency of the model and accelerates the convergence process.

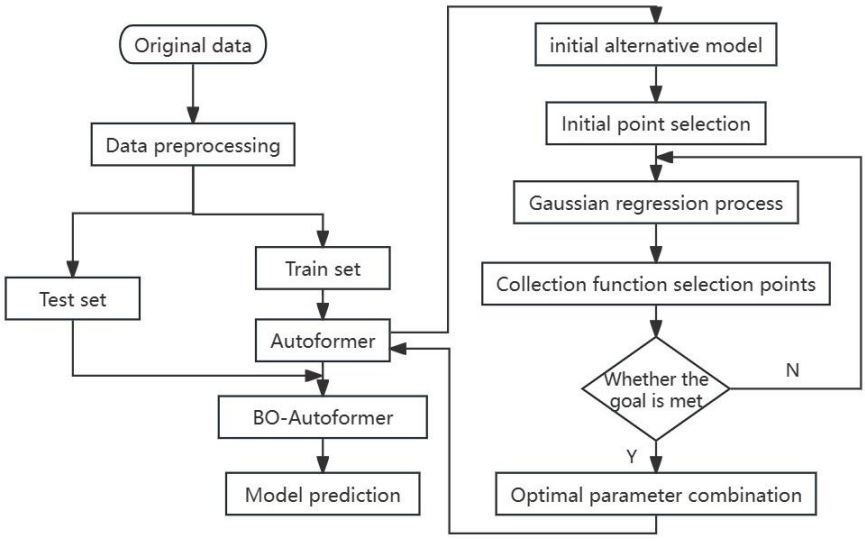

Fig. 2. Workload prediction flow chart based on BO-Autoformer.

Step 4: The Autoformer model adjusted by Bayesian optimization predicts the test set data.

V. Results and Discussion

A. Experimental Environment Configuration

The entire combined model is written in Python3.11 and implemented based on Pytorch and is finally executed on Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.71 GHz.

B. Dataset and Data Process

The experiment conducted in this study utilizes the Google Cluster Trace dataset, a real-world workload dataset released by Google in 2011. This extensive dataset comprises over 40 million tasks, encompassing workload information from approximately 12,500 machines over a span of 29 days. It encompasses various attributes such as job ID, task index, machine ID, CPU usage, and memory usage. This study focuses on CPU usage as the primary load information. The original data is sampled every five minutes, resulting in a total of 8333 sampling points.

To ensure the rigor and reliability of the findings, the dataset was carefully divided into three distinct subsets: training, validation, and testing. Specifically, the initial 4986 sets of data are designated as the training set, representing approximately 60% of the total dataset. This ensures that the model is adequately trained on a substantial portion of the available data. Subsequently, the following 1668 sets of data serve as the validation set, accounting for 20% of the dataset. The validation set is utilized to monitor the model's performance during training, aiding in hyperparameter tuning and preventing overfitting. Finally, the remaining 1679 sets of data comprise the testing set, also constituting 20% of the total dataset. The testing set enables us to evaluate the model's generalization ability on unseen data, providing an unbiased assessment of its performance.

This systematic approach ensures a balanced allocation of data for training, validation, and testing, allowing us to comprehensively assess the performance of the model and ensure its reliability in real-world scenarios.

1) Missing value process: Given the extensive data sample size and high statistical frequency inherent in the Google Cluster Trace dataset, it is inevitable that occasionally, specific reasons may lead to the omission of individual data points. To address this issue, this article employs the linear regression fitting interpolation method as a robust approach to fill in the missing values. This method ensures that the missing data are accurately and reliably estimated, minimizing any potential biases or distortions in the subsequent analysis.

2) Data normalization: Data normalization serves as a crucial step in enhancing the training effectiveness of neural networks. It significantly accelerates the process of locating optimal solutions during training, thereby improving the overall performance of the network. This study adopts the minimum-maximum normalization method, which effectively scales the input values to fall within the range of 0 to 1. This normalization approach ensures that the data is appropriately scaled, minimizing potential biases and enhancing the convergence speed of the training process.

C. Evaluation Index

To assess the precision of the proposed model accurately, several key evaluation metrics were selected, including Mean Square Error (MSE), Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percent Error (MAPE). These metrics provide comprehensive insights into the model's performance. The formulas for each evaluation index are as shown in Eq. (13) to Eq. (16):

$$e_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{13}$$

$$e_{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{14}$$

$$e_{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{15}$$

$$e_{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\% \tag{16}$$

In the aforementioned equations, $\hat{y}_i$ and y_i, represent the predicted load value and the actual load value at time i respectively. Additionally, n signifies the total count of test samples. It is noteworthy that as these evaluation indicators approach zero, the prediction performance of the model improves significantly.

D. Discussion

The main finding of this study is that the Autoformer model was successfully applied to the field of cloud computing workload prediction, and the model's hyperparameter combination was optimized through Bayesian optimization technology. This discovery not only demonstrates the great potential of the Autoformer model in the field of cloud computing but also proves the effectiveness of Bayesian optimization technology in hyperparameter tuning. Through this method, the accuracy and efficiency of cloud computing workload prediction are improved, and new ideas are provided to solve resource management problems in cloud computing.

The significance of this study is profound, both theoretically and practically. At the theoretical level, this study combines the Autoformer model and Bayesian optimization technology to propose a new cloud computing workload prediction method, which enriches the research content in the field of cloud computing resource management. At a practical level, this approach promises to augment the efficiency and precision of cloud computing resource management, leading to reduced operational costs and improved service quality. For cloud computing service providers, this means being able to better respond to workload changes, optimize resource allocation, and improve resource utilization. For cloud computing users, it means they can obtain a more stable and efficient service experience.

E. Experimental Results

1) Artificiality: In the context of cloud workload data prediction, a complex network model may often give rise to issues such as overfitting, inadequate training, and underfitting. Therefore, it is imperative to meticulously address the complexity of the network model when selecting hyperparameters. In the experiments, we manually adjusted the hyperparameters and conducted multiple comparisons of prediction results. The selected hyperparameters, as presented in Table II, were carefully chosen to balance the model's complexity with its predictive capabilities.

2) Bayesian optimization: In the Autoformer model, the sequence length optimization range was set to [6, 48] with a step size of 6, allowing for the exploration of various sequence lengths and their impact on model performance. The model dimension optimization range encompassed [128, 256, 512, 1024], permitting the identification of an appropriate balance between model complexity and predictive accuracy. Additionally, the batch size optimization range was established as [12, 24, 32, 48], enabling the investigation of the effect of batch size on training efficiency and stability. Furthermore, the optimization range for the middle layer dimension of the feedforward network was set to [512, 1024, 2048], allowing for the exploration of different network depths. The number of attention heads was optimized within the range of [1, 8], exploring the trade-off between attention granularity and computational complexity. Moreover, the encoder and decoder layers were optimized over the range of [1, 5], studying the impact of model depth on prediction performance. Finally, the optimization ranges for the attention factor and regularization coefficient were set to [1, 5] and [0.0, 0.5], respectively, facilitating the adjustment of model sensitivity and generalization ability.

TABLE II. HYPERPARAMETER COMBINATIONS DETERMINED BY ARTIFICIALITY

Hyperparameters	Meaning	Value
seq_len	The maximum length of the input sequence processed by the model at each time	18
d_model	Model embedding dimension, also known as hidden layer size	128
batch_size	Number of samples processed by the model in each iteration	32
d_ff	Internal Dimensions in Feedforward Neural Networks	1024
n_heads	Number of heads in multi-head self-attention	8
e_layers	Number of encoder layers	3
d_layers	Number of decoder layers	2
factor	Controlling the number of basis functions in the attention mechanism	2
dropout	The probability of dropping at random	0.05

Utilizing the Bayesian optimization algorithm, this paper conducted a meticulous hyperparameter search for the Autoformer model. Through iterative evaluations of the objective function, the algorithm identifies the hyperparameter combinations that yield minimal loss, thereby maximizing prediction accuracy. The optimal hyperparameters, as summarized in Table III, represent the most effective configuration for the Autoformer model in terms of balancing model complexity, training efficiency, and predictive performance.

As presented in Table III, the hyperparameter values obtained following Bayesian optimization are as follows: sequence length of 18, model dimension of 512, batch size of 32, intermediate layer dimension of the feedforward network set to 2048, eight attention heads, two encoder layers, one decoder layer, an attention factor of 3, and a regularization coefficient of 0.05. Through multiple runs, this paper observed the convergence speed and computational efficiency of the neural network. To ensure the convergence of experimental errors, the maximum traversal count was fixed at 10. This optimized configuration enables the Autoformer model to achieve superior predictive performance while balancing computational demands.

3) Compare and Analysis: Table IV comprehensively summarizes the evaluation metrics of the Autoformer model before and after the application of Bayesian optimization. Employing the optimal hyperparameter combination, the study tested the performance of the Autoformer model and conducted a comparative analysis between the original Autoformer and the BO-Autoformer, using real-world data. Fig. 3 provides a visual representation of the improvement achieved result through Bayesian optimization.

As evident from Table IV, the BO-Autoformer model exhibits slight improvements in various evaluation metrics compared to the original Autoformer. Specifically, the mean squared error (MSE) is reduced by 0.82%, the root mean squared error (RMSE) is decreased by 0.41%, the mean absolute error (MAE) is lowered by 0.55%, and the mean absolute percentage error (MAPE) is diminished by 0.59%. These results demonstrate the effectiveness of Bayesian optimization in enhancing the predictive accuracy of the Autoformer model.

One can clearly observe from Fig. 3 that the predicted value obtained by the BO-Autoformer model is close to the real value, and the load curve value is relatively close. The findings indicate that the BO-Autoformer model can predict cloud workload data better than Autoformer.

TABLE III. HYPERPARAMETER COMBINATIONS DETERMINED BY BAYESIAN OPTIMIZATION

Hyperparameters	Parameter adjustment range	Value
seq_len	（6,48）	18
d_model	[128, 256, 512, 1024]	512
batch_size	[12, 24, 32, 48]	32
d_ff	[512, 1024, 2048]	2048
n_heads	（1, 8）	8
e_layers	（1, 5）	2
d_layers	（1, 5）	1
factor	（1, 5）	3
dropout	（0.0, 0.5）	0.05

TABLE IV. EVALUATION INDICATORS BEFORE AND AFTER BAYESIAN OPTIMIZATION

Model	MSE	RMSE	MAE	MAPE
Autoformer	0.003046	0.055190	0.043399	0.193858
BO-Autoformer	0.003021	0.054961	0.043159	0.192713

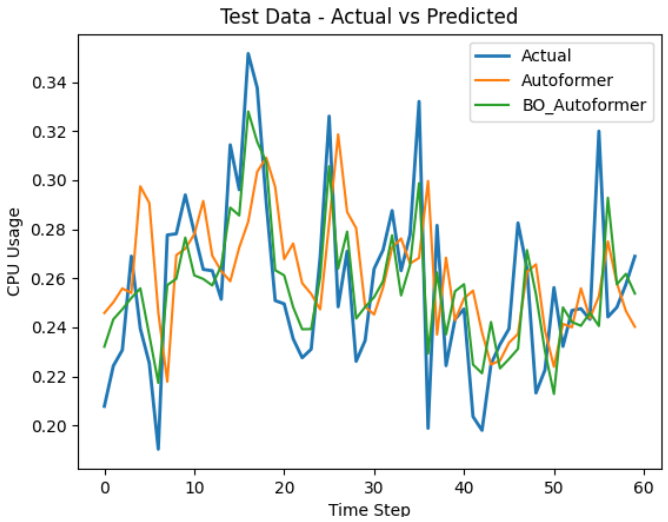

Fig. 3. Comparison of predicted values and actual values before and after Bayesian optimization.

The optimal hyperparameter combinations obtained by Bayesian optimization were applied to the four benchmark models: DLinear[29]、NLinear[29]、Informer and Reformer, and a comprehensive comparison among them. The evaluation metrics of the various models are summarized in Table V, while the prediction outcomes are graphically presented in Fig. 4, providing a clear and concise visualization of the comparative performance.

As indicated in Table V, it is evident that the MSE of the BO-Autoformer model is reduced to 0.003021, which is more stable than the other four methods. In comparison to the remaining four approaches, the RMSE has diminished by 16.50%, 15.44%, 2.33%, and 1.15% respectively, the span of the discrepancy between the true and predicted values has narrowed, and the prediction will be more reasonable. Compared with the other four methods, the model MAE has diminished by 16.97%, 15.15%, 0.78%, and 1.24% respectively. Combined with the prediction trend, it is apparent that the precision of predictions has risen. In relation to the other three methodologies, there is a decrease in the MAPE value by 16.09%, 11.09%, and 1.78% respectively, and the quality of the model has been slightly improved. These findings collectively demonstrate the superiority of the BO-Autoformer model in terms of prediction accuracy and stability.

TABLE V. EVALUATION INDICATORS OF BASELINE MODELS

Model	MSE	RMSE	MAE	MAPE
DLinear	0.004333	0.065825	0.051978	0.229670
NLinear	0.004225	0.064997	0.050863	0.216753
Informer	0.003167	0.056272	0.043499	0.184837
Reformer	0.003091	0.055598	0.043701	0.196201
BO-Autoformer	0.003021	0.054961	0.043159	0.192713

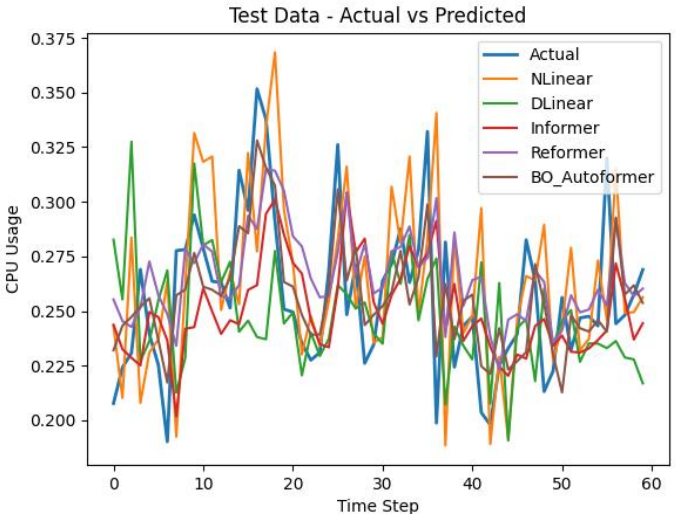

Fig. 4. Comparison of predicted values and actual values of the baseline models.

As demonstrated in Fig. 4, the fitting curve of the Bayesian optimized Autoformer model is closer to the true value than other models, and the fitting effect is the best. Consequently, the BO-Autoformer model proposed in this study demonstrates significantly higher prediction accuracy when compared to several alternative models, thereby underscoring its effectiveness and reliability in the domain of concern.

In the scenario where the input sequence spans 18-time steps (equating to 90 minutes), the study conducted a comparative analysis across various prediction horizons, specifically 1-time step (5 minutes), 6-time steps (30 minutes), 12-time steps (60 minutes), and 18-time steps (90 minutes). The comparative outcomes were systematically compiled in Table VI, facilitating a comprehensive evaluation against other models.

Table VI clearly illustrates that the MSE and MAE error coefficients associated with the 5-minute, 30-minute, 60-minute, and 90-minute predictions of the BO-Autoformer model are predominantly lower than those of other models,

thus achieving optimal performance. Therefore, the Autoformer model demonstrates notable superiority over other models in both short-term and long-term prediction capabilities, highlighting its efficacy and reliability across various prediction horizons.

4) Discussion of limitations: Despite the encouraging results of this study, several limitations and potential issues remain. Firstly, the validation was conducted using only a single dataset, which may not fully capture the performance of the BO-Autoformer model across diverse scenarios. This limitation may cause our evaluation of model performance to be overly optimistic or one-sided. Therefore, future research should expand the dataset scope to better assess the model's generalization ability. Secondly, the Bayesian optimization algorithm used to find the optimal hyperparameter combination can be computationally intensive, making it unsuitable for application scenarios requiring high real-time performance. This limitation limits the application of the BO-Autoformer model in scenarios such as online learning. Therefore, future research should explore more efficient optimization algorithms to enhance the training speed and prediction efficiency of the model.

TABLE VI. EVALUATION INDICATORS OF BASELINE MODELS UNDER DIFFERENT PREDICTION LENGTHS

Model/prediction length	5min		30min		60min		90min	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
DLinear	0.004333	0.051978	0.005220	0.056784	0.006236	0.062536	0.006991	0.066337
NLinear	0.004225	0.050863	0.004734	0.053796	0.004969	0.055146	0.005497	0.057858
Informer	0.003167	0.043499	0.003730	0.047181	0.003975	0.049355	0.004357	0.051340
Reformer	0.003091	0.043701	0.003551	0.046383	0.003956	0.049906	0.004659	0.054540
BO-Autoformer	0. 003021	0.043159	0.003503	0.046312	0.003794	0.048110	0.003965	0.049038

VI. CONCLUSION AND FUTURE WORK

Given the intricate sequential patterns and complexities inherent in cloud workload, accurate prediction of the workload holds paramount importance for successful cloud computing resource management. To address the prevailing challenges of limited prediction accuracy and challenging hyperparameter tuning in cloud workload prediction, this paper introduces the BO-Autoformer model, a fusion of the Autoformer model and Bayesian optimization techniques. Through rigorous experimental validation, the BO-Autoformer model was found to significantly outperform the traditional Autoformer model, achieving a reduction in MSE and MAE by 0.82% and 0.55% respectively, thereby enhancing prediction accuracy. By comparing with 4 baseline models, it is found that this model promises extensive application potential in both short-term and long-term load prediction.

Future research should not only be satisfied with the existing prediction accuracy but should continue to explore new optimization paths to achieve further improvement in the performance of prediction models. In addition, designing a reasonable virtual machine consolidation strategy based on the prediction results to realize the efficient utilization of cloud resources is also an important research direction in the future.

ACKNOWLEDGMENT

This work was supported by the Science and Research Project of Harbin University of Commerce (2019DS032).

REFERENCES

[1] Arianyan E, Taheri H, Sharifian S. Novel heuristics for consolidation of virtual machines in cloud data centers using multi-criteria resource management solutions[J]. The Journal of Supercomputing, 2016, 72(2): 688-717.

[2] Rong H, Zhang H, Xiao S, et al. Optimizing energy consumption for data centers[J]. Renewable and Sustainable Energy Reviews, 2016, 58: 674-691.

[3] Uddin M, Shah A, Alsaqour R, et al. Measuring Efficiency of Tier Level Data Centers to Implement Green Energy Efficient Data Centers[J]. 2013.

[4] Avgerinou M, Bertoldi P, Castellazzi L. Trends in Data Centre Energy Consumption under the European Code of Conduct for Data Centre Energy Efficiency[J]. Energies, 2017, 10(10): 1470.

[5] Wu H, Xu J, Wang J, et al. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting[J]. 35th Conference on Neural Information Processing Systems, 2021.

[6] Yazhou Hu, Bo Deng, Fuyang Peng, et al. Workload prediction for cloud computing elasticity mechanism[C]//2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA). Chengdu, China: IEEE, 2016: 244-249.

[7] Jiang Y, Perng C shing, Li T, et al. ASAP: A Self-Adaptive Prediction System for Instant Cloud Resource Demand Provisioning[C]//2011 IEEE 11th International Conference on Data Mining. Vancouver, BC, Canada: IEEE, 2011: 1104-1109.

[8] Tirado J M, Higuero D, Isaila F, et al. Predictive Data Grouping and Placement for Cloud-Based Elastic Server Infrastructures[C]//2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. Newport Beach, CA, USA: IEEE, 2011: 285-294.

[9] Aditya Satrio C B, Darmawan W, Nadia B U, et al. Time series analysis and forecasting of coronavirus disease in Indonesia using ARIMA model and PROPHET[J]. Procedia Computer Science, 2021, 179: 524-532.

[10] Tang X, Liao X, Zheng J, et al. Energy efficient job scheduling with workload prediction on cloud data center[J]. Cluster Computing, 2018, 21(3): 1581-1593.

[11] Xie Y, Jin M, Zou Z, et al. Real-Time Prediction of Docker Container Resource Load Based on a Hybrid Model of ARIMA and Triple Exponential Smoothing[J]. IEEE Transactions on Cloud Computing, 2020, 10(2): 1386-1401.

[12] Melhem S B, Agarwal A, Goel N, et al. Markov Prediction Model for Host Load Detection and VM Placement in Live Migration[J]. IEEE Access, 2018, 6: 7190-7205.

[13] Huang P, Ye D, Fan Z, et al. Discriminative Model for Google Host Load Prediction with Rich Feature Set[C]//2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. Shenzhen, China: IEEE, 2015: 1193-1196.

[14] Rossi A, Visentin A, Prestwich S, et al. Uncertainty-Aware Workload Prediction in Cloud Computing[M]. arXiv, 2023[2023-11-02]. http://arxiv.org/abs/2303.13525.

[15] Liu C, Liu C, Shang Y, et al. An adaptive prediction approach based on workload pattern discrimination in the cloud[J]. Journal of Network and Computer Applications, 2017, 80: 35-44.

[16] Borkowski M, Schulte S, Hochreiner C. Predicting cloud resource utilization[C]//Proceedings of the 9th International Conference on Utility and Cloud Computing. Shanghai China: ACM, 2016: 37-42.

[17] Duggan M, Mason K, Duggan J, et al. Predicting host CPU utilization in cloud computing using recurrent neural networks[C]//2017 12th International Conference for Internet Technology and Secured Transactions (ICITST). Cambridge: IEEE, 2017: 67-72.

[18] Guo Y, Yao W. Applying gated recurrent units pproaches for workload prediction[C]//NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium. Taipei: IEEE, 2018: 1-6.

[19] Golshani E, Ashtiani M. Proactive auto-scaling for cloud environments using temporal convolutional neural networks[J]. Journal of Parallel and Distributed Computing, 2021, 154: 119-141.

[20] Qiu F, Zhang B, Guo J. A deep learning approach for VM workload prediction in the cloud[C]//2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD). Shanghai, China: IEEE, 2016: 319-324.

[21] Guo W, Ge W, Lu X, et al. Short-Term Load Forecasting of Virtual Machines Based on Improved Neural Network[J]. IEEE Access, 2019, 7: 121037-121045.

[22] Xu M, Song C, Wu H, et al. esDNN: Deep Neural Network Based Multivariate Workload Prediction in Cloud Computing Environments[J]. ACM Transactions on Internet Technology, 2022, 22(3): 1-24.

[23] Vaswani A, Shazeer N, Parmar N, et al. Attention is All you Need[J]. 31st Conference on Neural Information Processing Systems, 2017.

[24] Li S, Jin X, Xuan Y, et al. Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting[M]. arXiv, 2020[2023-11-11]. http://arxiv.org/abs/1907.00235.

[25] Kitaev N, Kaiser Ł, Levskaya A. Reformer: The Efficient Transformer[M]. arXiv, 2020[2023-11-11]. http://arxiv.org/abs/2001.04451.

[26] Zhou H, Zhang S, Peng J, et al. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2021, 35(12): 11106-11115.

[27] Snoek J, Larochelle H, Adams R P. Practical Bayesian Optimization of Machine Learning Algorithms[J]. 2012.

[28] Wu J, Chen X Y, Zhang H, et al. Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization[J]. 2019, 17(1).

[29] Cho H, Kim Y, Lee E, et al. Basic Enhancement Strategies When Using Bayesian Optimization for Hyperparameter Tuning of Deep Neural Networks[J]. IEEE Access, 2020, 8: 52588-52608.

[30] Abbasimehr H, Paki R. Prediction of COVID-19 confirmed cases combining deep learning methods and Bayesian optimization[J]. Chaos, Solitons & Fractals, 2021, 142: 110511.

[31] Jin X B, Zheng W Z, Kong J L, et al. Deep-Learning Forecasting Method for Electric Power Load via Attention-Based Encoder-Decoder with Bayesian Optimization[J]. Energies, 2021, 14(6): 1596.

[32] Zeng A, Chen M, Zhang L, et al. Are Transformers Effective for Time Series Forecasting?[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2023, 37(9): 11121-11128.