IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 11, 2020, accepted June 18, 2020, date of publication June 23, 2020, date of current version July 2, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.3004370*

# An Accurate Ensemble Forecasting Approach for Highly Dynamic Cloud Workload With VMD and R-Transformer

**SHAOJING ZHOU, JINGUO LI, (Member, IEEE), KAI ZHANG, MI WEN, (Member, IEEE), AND QIJIE GUAN**

College of Computer Science and Technology, Shanghai University of Electric Power, Shanghai 200090, China

Corresponding author: Jinguo Li (lijg@shiep.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61702321, Grant 61802248, and Grant U1936213, and in part by the Shanghai Education Development Foundation and Shanghai Municipal Education Commission through the Chenguang Program under Grant 18CG62.

**ABSTRACT** To efficiently manage the cloud resources, improve the quality of service, and avoid the violations of Service-Level Agreement (SLA) agreements, it is very important to make accurate forecast for cloud workload. Prior works concerning cloud workload forecasting are mainly designed based on Recurrent Neural Networks (RNN). However, when it comes to a highly-dynamic cloud workload scenario where resource utilization changes faster and more frequently, these RNN-based methods are not effective in obtaining the linear and non-linear relationships and cannot give accurate forecast, because classic RNN has the problem of vanishing gradient. To address this issue, we propose an Ensemble Forecasting Approach (EFA) for highly-dynamic cloud workload by applying Variational Mode Decomposition (VMD) and R-Transformer. Specifically, to decrease the non-stationarity and high randomness of highly-dynamic cloud workload sequences, we decompose the workload into multiple Intrinsic Mode Functions (IMFs) by VMD. The IMFs are then imported into our ensemble forecasting module based on R-Transformer and Autoregressive model, in order to capture long-term dependencies and local non-linear relationship of workload sequences. The effectiveness and adaptability of proposed EFA is verified on real-world workload from Google and Alibaba cluster traces. Moreover, the performance evaluation results show that the EFA performs higher forecasting accuracy than prior related works over various forecasting time lengths for highly-dynamic cloud workload.

**INDEX TERMS** Workload forecasting, cloud computing, deep learning, variational mode decomposition.

## I. INTRODUCTION

One of the main outstanding properties of cloud computing system is elasticity [1]. The elasticity implies that the system can automatically provision or de-provision resources to adapt to workload changes [2]. An efficient resource management scheme can proactively identify the possible resource usage patterns to predict the future workload of the cloud center and provide the required resources [3]. However, conducting proactive resource management is not easy, for instance, some hot events may attract lots of traffic in a short period in social networks. If the cloud service providers cannot provide enough resources in time, i.e., under-provisioning, it would reduce the Quality of Service (QoS) and violate the Service-Level Agreement (SLA), which would lead to customer churn. On the other hand, if the cloud service provider offers excess available resources all the time, i.e., over-provisioning, it would waste a lot of energy, and incur extra costs on network traffic, device cooling and maintenance [4]. Therefore, accurate workload forecasting is a key factor in implementing efficient proactive resource management schemes and quickly allocating resources to what users need [3].

Moreover, for large-scale cloud centers, the workload patterns are very diverse and random [5], which makes it challenging to compute accurate forecast results on workload. Analysis of the Alibaba cluster dataset shows that the CPU usage of the entire cluster changes from 5% to 80% in a day

The associate editor coordinating the review of this manuscript and approving it for publication was Quan Zou.

with high fluctuations [6]. Similarly, in the Google Cloud Data Center, changes in CPU, memory, and other resources are highly dynamic and random [7]. Therefore, an accurate workload forecasting approach is required. The approach can effectively capture the linear and non-linear correlations of workload, and can be adapted to highly variable workload. In addition, corresponding to the randomness of the workload, the characteristics of the original workload data should be further analyzed and extracted. The goal of our research is to extract characteristics from the historical sequence of workload and accurately forecast future workload changes.

In recent years, there have been many studies on cloud workload forecasting [12]–[25]. However, most of these researches are based on traditional regression methods or classic machine learning methods. In general, they have accurate results only in specific scenarios, such as when workload sequences have significant periodicity or regularity. Specifically, the traditional back-propagation neural network has considerable ability to capture the non-linear features of the sequence, but it does not make full use of the correlation between neurons [8], and it is mostly used for low variance cloud workload. In addition, in the cloud workload forecasting scenario, deep learning methods based on Recurrent Neural Networks (RNN) [9] have become popular due to its excellent ability to process sequence data. RNN is very suitable to forecast random workload, but traditional RNNs cannot accurately capture the dependency information of long sequences because of the vanishing gradient problem. The vanishing gradient means that when RNN is used to train long-sequence data, the weights of the previous neural units in the network cannot be updated, which eventually leads to the failure of network training [10], [11]. Variants of RNN, such as Long Short-Term Memory network (LSTM) [12] and Gated Recurrent Unit (GRU) [13], have been proposed to solve this problem. However, these RNN-based deep learning methods cannot give accurate forecast results when it comes to the highly-dynamic cloud center workload.

To address the above mentioned problems, we proposed an Ensemble Forecasting Approach (EFA) for highly-dynamic cloud workload based on Variational Mode Decomposition (VMD) and R-Transformer. The main contributions of this paper are summarized as follows:

- In order to decrease the non-stationarity of highly-dynamic cloud workload sequences, the Variational Mode Decomposition is used to preprocess workload data sequence and decompose it into multiple Intrinsic Mode Functions (IMFs) with good characteristics. In particular, nonlinear features in the original sequence are also effectively extracted.
- In order to overcome the shortcomings of RNN, IMFs will be input to an ensemble cloud workload forecasting module based on R-Transformer and Autoregressive model. This module uses LocalRNN to obtain the local non-linear relationship of the sequences, and captures long-term dependencies through the multi-head attention mechanism. Moreover, The Autoregressive model can obtain the linear relationship of workload and improve the robustness and forecasting accuracy of proposed EFA.
- To demonstrate the effectiveness of the proposed EFA, we apply this methodology to real-world cloud center workload datasets from Google's cluster and Alibaba's cluster. The experimental results show that the proposed approach can capture local and global information and achieve higher forecasting accuracy than prior related works over various forecasting time lengths.

The rest of the paper is organized as follows. Section II reviews the most relevant related works to cloud workload forecasting. Section III describes the system architecture of EFA. In Section IV, the proposed EFA is discussed in detail. Performance evaluation is presented in Section V. Finally, Section VI concludes this paper.

## II. RELATED WORK

At present, there are many researches on workload forecasting in cloud computing, which can be divided into traditional regression models, classical machine learning methods and deep learning models. They all extract patterns from historical workload data to forecast future changes.

### A. TRADITIONAL REGRESSION METHODS

Traditional regression methods mainly include Autoregressive (AR), Moving Average (MA), Autoregressive Moving Average (ARMA), Autoregressive Integrated Moving Average (ARIMA). Hu *et al.* [14] proposed an Autoregressive-based approach to forecast the workload, but the model is strictly linear in nature and lacks adaptability to workload in complex cloud environments. Amekraz and Hadi [15] used an adaptive ARMA model and validated it on a web dataset. However, this model is limited to cloud environments that have significant periodicity. Calheiros *et al.* [16] used a predictor based on the ARIMA to proactively configure virtual machine (VM) instances. They demonstrated ARIMA's ability to perform short-term forecasting and used the results to dynamically provision resources for SaaS applications. Saripalli *et al.* [17] proposed a two-step approach, including load trace (LT) and load forecasting (LF). The proposed cubic-spline LT can model the high fluctuations of the loads better than the other linear LTs based on Moving Average. In another work, Guo *et al.* [18] proposed a type-aware workload forecasting strategy, which dynamically determines the type of workload and relies on its type to adaptively switch forecasting algorithms.

Traditional regression methods make time series forecasting model less complex compared to other methods such as artificial neural networks. Although the model based on regression methods are simple, they are based on oversimplified assumptions (linear relationships) of workload [4]. Therefore, they are unable to capture nonlinear changes in workload.

### B. CLASSICAL MACHINE LEARNING METHODS

Classic machine learning technology has been widely used in many large and complex data-intensive fields, especially in the field of data processing and workload forecasting [19]. The classical machine learning methods mainly include Markov-based models, Bayesian models, support vector regression (SVR), decision tree, and traditional artificial neural networks. Ghobaei-Arani *et al.* [20] proposed a hybrid approach using the Markov Decision Process (MDP) model. The Markov decision process is present with finite states and transitions among states. Shyam and Manvi [21] proposed a Bayesian model to determine short and long-term virtual resource requirement of the CPU/memory intensive applications on the basis of workload patterns at several data centers in the cloud during several time intervals. Singh *et al.* [22] proposed an adaptive forecasting model called TASM by using linear regression, ARIMA, and SVR for web applications, and they proposed a workload classifier that can select models based on workload characteristics. Rahmanian *et al.* [23] proposed an ensemble cloud resources usage forecasting algorithm based on Learning Automata (LA) theory. The algorithm employed two methods, namely Single Window (SW) and Multiple Window (MW) for cloud resource forecasting.

Most traditional machine learning or classic artificial neural network workload forecasting methods do not require restrictive assumptions about the form of workload, and can extract non-linear characteristics of workload. However, they require workload with obvious regularity or trends to achieve accurate forecast because they rely mostly on heuristic algorithms.

### C. DEEP LEARNING METHODS

Over the past few years, researchers have applied several deep learning algorithms to forecast workload in the cloud, mainly including Recurrent Neural Network (RNN), Long Short-Term Memory network (LSTM), Convolutional Neural Network (CNN) and Deep Belief Network (DBN) [9]. Duggan *et al.* [24] used the classic RNN architecture to forecast the future workload of cloud data centers. It turned out that RNN can work well when coping with short-term dependencies. Zhu *et al.* [25] proposed a forecasting approach using attention-based LSTM encoder-decoder network. The method extracts the characteristics of historical workload data through an encoder. They integrate an attention mechanism in the decoder to obtain the weight of predictions at different historical time steps. [26] designed and proposed an improved LSTM-based model N-LSTM to solve the problem of virtual machine workload forecasting, and it could make forecast at irregular time intervals. Guo and Yao [27] proposed a method for workload prediction based on GRU [13]. The model can learn temporal patterns and long-term dependencies of large sequences of arbitrary length, and the model training time is shorter than that of LSTM. Zhang *et al.* [28] proposed an efficient deep learning model based on the canonical polyadic decomposition to forecast the cloud workload for industry informatics. They compressed the parameters significantly by converting the weight matrices to the canonical polyadic format, and designed a learning algorithm to train the parameters.

However, most deep learning algorithms, including LSTM and GRU, are designed based on RNN architecture. Although these algorithms are excellent at capturing non-linear information for cloud workload, they cannot adapt to workload with high dynamics and high random, which might lead to low accuracy and high computational complexity for workload forecasting.

## III. SYSTEM ARCHITECTURE

In order to accurately and efficiently adjust resources in a cloud environment, cloud service providers need to forecast workload changes in the future based on historical workload data. In response to the highly-dynamic workload, we propose the EFA in cloud systems. The details of EFA are shown in Fig. 1.

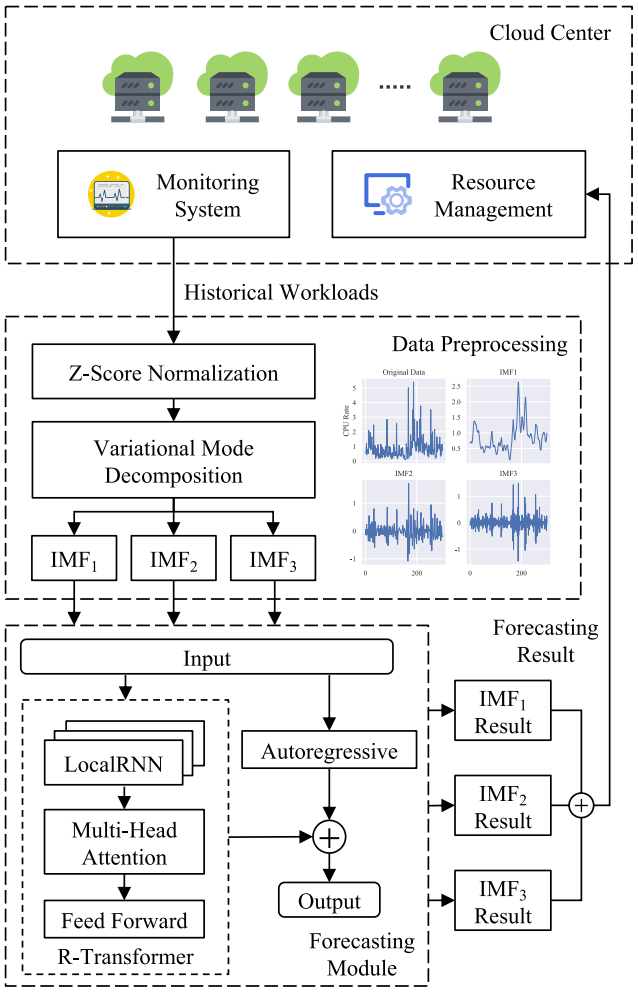



**FIGURE 1.** The architecture of proposed EFA in cloud systems. The data preprocessing part uses VMD to decompose the normalized historical workload, the forecasting module consists of R-Transformer and Autoregressive model.

First, the monitoring system in the cloud center records the workloads of the system in real time. In proposed EFA, these historical workloads are from the monitoring system. Then, historical data is processed using Z-score normalization and VMD, which will be explained in detail in Section IV-A. The results are considered as inputs to the ensemble forecasting module. Next, use the ensemble forecasting module composed of R-Transformer and Autoregressive model to forecast different IMF, and we will explain this process in Section IV-B. Finally, the forecasting results of each IMF are added to obtain the final forecasting result, and this result is input to the resource management system of the cloud center for resources allocation or recovery.

Generally, cloud workload data includes various indicators about the operating status of the system, like CPU and memory usage, disk space, disk I/O time, bandwidth, etc. CPU load can be regarded as the most important and limited resources in a computer system, and also the main bottleneck in cloud platforms [29]. Therefore, the industry regards CPU usage as a key factor in improving the resource allocation of cloud centers.

In this paper, we focus on CPU usage. In order to capture information from historical data, the historical CPU usage of the cloud center can be presented in the form of a time series $\vec{x} = (x_1, x_2, \ldots, x_t)$ which is a sequence of recorded values arranged in chronological order with constant time intervals, and the CPU usage record value at time $t$ is $x_t$. The EFA use sliding window forecasting method to predict future CPU usage $x_{t+h}$ based on $\vec{x}$, where $h$ is the forecasting length ahead of the current time stamp $t$. Moreover, the EFA predict the future CPU usage of $x_{t+h+n}$ based on $(x_{1+n}, x_{2+n}, \ldots, x_{t+n})$, $n \in \mathbb{R}^+$.

## IV. THE ENSEMBLE FORECASTING APPROACH FOR CLOUD WORKLOAD

This section presents the proposed EFA, an ensemble forecasting approach for cloud workload. In Section IV-A we describe the process of data preprocessing. In Section IV-B, the ensemble forecasting module is described in detail.

### A. DATA PREPROCESSING

In this section, details of data preprocessing is given as follows. We will detail the process of data preprocessing, including Z-score normalization and VMD.

#### 1) Z-SCORE NORMALIZATION

According to the actual highly-dynamic cloud workload, the value of CPU usage varies greatly in different time intervals, so the original time series datas need to be normalized. In addition, this can also accelerate the convergence of deep learning algorithms. Considering the large range of fluctuations in cloud workloads, such as in the Google cluster dataset, where most of the data values are in a small range, using Min-Max normalization would cause these large numbers of smaller values to be concentrated in some small range, which is not conducive to training and convergence of the forecasting module. Therefore, in this part, Z-score normalization is used to normalize the original sequences. The processed data conforms to the standard normal distribution, that is, the mean is 0 and the standard deviation is 1. The formula for Z-score normalization is below:

$$x' = \frac{x - mean(X)}{\sigma}, \tag{1}$$

where $X$ is the set of $x$, $mean(X)$ is the mean value of $X$ and $\sigma$ is the standard deviation $\sqrt{\mathrm{E}((X - \mathrm{E}(X))^2)}$ of $X$.

#### 2) VARIATIONAL MODE DECOMPOSITION

The VMD is a non-recursive signal processing algorithm, the purpose of VMD is to decompose an input signal into $k$ discrete number of sub-signals (modes), where each mode has limited bandwidth in spectral domain [30]. Each mode can be compacted around a center pulsation $\omega_k$ determined during the decomposition process. To obtain the bandwidth of a one dimension signal, three steps should be fulfilled [30]: Step 1, for each mode, adopt Hibert transform to obtain a unilateral frequency spectrum. Step 2, for each mode, transfer the mode's frequency spectrum to the baseband by applying an exponential tuned to the respective estimated center frequency. Step 3, for each mode, estimate the bandwidth by utilizing the Gaussian smoothness of the demodulated signal. The constrained variational problem can be given as follows [30]:

$$\begin{aligned} &\min_{u_k, \omega_k} \left\{ \sum_k \left\| \partial_t \left[ \left( \delta(t) + \frac{j}{\pi t} \right) * u_k(t) \right] e^{-j\omega_k t} \right\|_2^2 \right\}, \\ &\text{s.t.} \sum_k u_k = f, \end{aligned} \tag{2}$$

where $\partial$ is the Dirac distribution, $f$ is the original signal and $u$ is its mode, $u_k$ denotes the k-th mode, $\omega_k$ is the center frequency, $t$ is time script, and $*$ is the convolution operator. The mode $u$ with high-order $k$ represents low frequency components.

The components obtained after using VMD are often called Intrinsic Mode Function (IMF) signals. In the context of this paper, they can also be seen as a set of different time series. Compared with the original time series, the instability of IMFs is reduced, and the information contained in each IMF covers different parts of the original time series [31].

In the proposed EFA, VMD is used to decompose highly-dynamic cloud workload data into stable and predictable IMFs. Similar to [32] and [33], each IMF can be regarded as a new time series. Usually the first IMF contains the low-frequency part of the original workload sequence, which can be regarded as a smooth trend change of the original sequence. The high-frequency part is included in the remaining IMF, which can accurately identify small details of the original sequence. Each IMF is independently input into the ensemble forecasting module, which will output the forecasting result of this IMF. The forecasting results of these IMFs are added to obtain the final workload forecast result.

### B. ENSEMBLE WORKLOAD FORECASTING MODULE

In this part, we continue to describe the proposed ensemble forecasting module in detail, which can make accurate forecast for the IMFs from the previous part.

#### 1) R-TRANSFORMER

Models based on RNN and its variants such as LSTM are widely used in the field of time series forecasting. However, these models cannot capture long-term dependencies well and cannot perform parallel calculations on sequences. Transformer [34] has been proposed and has proven to be extremely efficient at capturing long-term dependencies in various sequence modeling tasks especially in NLP [35]. Nevertheless, standard Transformer is not very suitable for time series forecasting, this is because it highly relies on position embedding for solving the loss of position sequence information, and it also lacks necessary components to model local structures in sequences, which can make the model prone to anomalies in time series [36], [37]. Inspired by [36], we use R-Transformer to obtain local and global information of time series. R-Transformer takes the advantages of RNN and multi-head attention mechanism while avoiding their respective shortcomings.

The architecture of R-Transformer is shown in Fig. 2. The R-Transformer is introduced in detail from three parts: LocalRNN layer, Multi-Head Attention layer and Feed-forward layer.

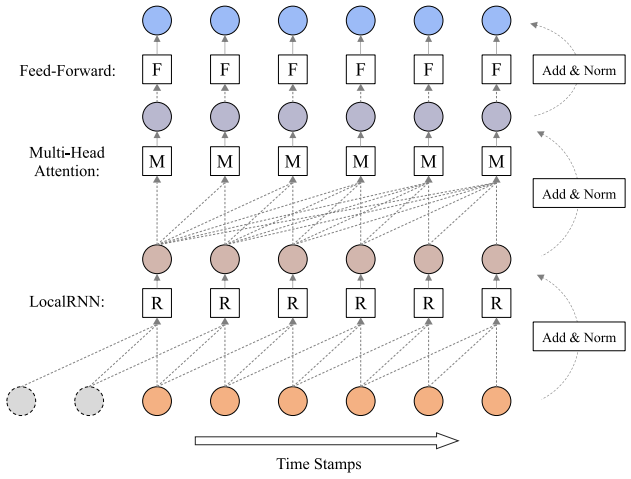



**FIGURE 2.** The architecture of one layer of R-Transformer. In particular, LocalRNN layer captures the local short-term dependencies; Multi-head attention layer captures the global long-term dependencies; Feed-forward layer performs non-linear feature transformation.

**Step 1:** LocalRNN reorganizes the original long workload sequence into many short sequences, so these short sequences contain only local information. These local sequences are processed independently and identically using RNN that share weights. Specifically, LocalRNN constructs a local windows of size $M$ for each target position, so that the local window includes $M$ consecutive positions and ends at the target position, and each window contains only local short-term dependencies. It is worth noting that $M$ is always smaller than the length of the original long sequence. Fig. 3 shows how LocalRNN works.

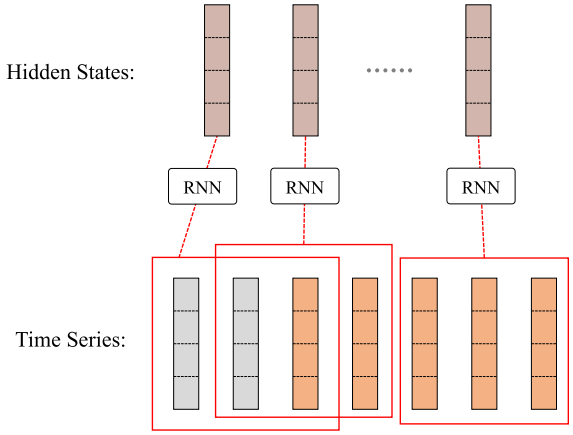



**FIGURE 3.** LocalRNN obtains local correlation information, it only operates on positions within a local window.

Concretely, given the positions $(x_{t-M+1}, x_{t-M+2}, \ldots, x_t)$ of local sequences of length $M$. LocalRNN processes these short sequences and outputs $M$ hidden states, then uses the last hidden states as a feature of the local sequence:

$$h_t = \text{LocalRNN}(x_{t-M+1}, x_{t-M+2}, \ldots, x_t), \tag{3}$$

where $t$ is the target time stamp, and RNN represents any RNN unit, such as LSTM and GRU. For an entire long workload sequence, to ensure that the hidden state representation of each time step is obtained and does not contain any future information, LocalRNNLayer first performs zero padding of length $M-1$ before the start of the sequence [36]. Then LocalRNN slides each window and outputs a hidden representation sequence containing local information:

$$h_1, h_2, \ldots, h_T = \text{LocalRNNLayer}(x_1, x_2, \ldots, x_T), \tag{4}$$

where $T$ denotes the length of input sequence. Then, the representation of local hidden states is input to multi-head attention layer to capture long-term global dependencies.

**Step 2:** According to recent works, the multi-headed attention mechanism is very effective at learning the long-term dependencies of sequences, as it can establish a direct connection between each pair of positions. The multi-head attention mechanism relies on scaled dot-product attention [34]:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V, \tag{5}$$

among them, $Q$, $K$ and $V$ are the matrix of query, key and value, and $d_k$ is the key dimensionality. The queries, keys and values are from the output of LocalRNN layer.

The multi-head attention mechanism obtains $h$ different representations of $(Q, K, V)$ respectively, $h$ can also be regarded as the number of heads, then calculates a scaled dot-product attention of each representation, and finally concatenates the results. Specifically, the current representations

$h_1, h_2, \ldots, h_T$ is input into the multi-head attention layer, and the new representation $u_t$ is calculated as follows:

$$\begin{aligned} u_t &= \text{MultiHeadAttention}(h_1, h_2, \ldots, h_T) \\ &= \text{Concatenation}(head_1(h_T), head_2(h_T), \\ &\quad \ldots, head_i(h_T), \ldots, head_h(h_T))W^o, \end{aligned} \tag{6}$$

where $head_i(h_T)$ is the result of $i^{th}$ attention head:

$$head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V), \tag{7}$$

where the $W^o$ and $W_i$ are parameter matrices and each attention head $head_i$ has its own mapping matrices $W_i$. As can be seen from Eq. (6), let $h_T$ participate in all past positions to get each $head_i$, so any long-term dependencies can be captured. And different heads can focus on obtaining different correlations.

**Step 3:** The layer behind the multi-head attention layer is the feed-forward layer, which is used to linearly transform features. Feed-forward layer is a fully connected layer, and it is applied to each position separately and identically. It consists of two linear transformations and uses a ReLU activation function between them. Feed-forward layer is defined as follows:

$$m_t = \text{FeedForward}(u_t) = \max(0, u_t W_1 + b_1)W_2 + b_2, \tag{8}$$

where $m_t$ is the output of the feed-forward layer, $W_1$ and $W_2$ are two different parameter matrices, $b_1$ and $b_2$ are the biases.

Finally, according to [34], a residual [38] and layer-norm [39] connection is added between all sub-layers of R-Transformer.

#### 2) AUTOREGRESSIVE MODEL

Due to the non-linear nature of both Recurrent and Multi-Head Attention components, the scale of neural network model output is not sensitive to the scale of input. And in cloud systems, the scale of workload constantly changes in a non-periodic manner, which greatly reduces the forecasting accuracy of forecasting models. To address the shortcoming, we use a mixture of linear and non-linear components as the final forecasting result. In our forecasting module, the classic Autoregressive model [40] is used as the linear component, and the Autoregressive model is formulated as follows:

$$l_t = \sum_{k=0}^{T-1} W_k^{ar} x_{t-k} + b^{ar}, \tag{9}$$

where $l_t$ denote the forecasting result of Autoregressive component, $W^{ar}$ is the parameters matrix, $T$ is the length of input window.

The final forecast result of the forecasting module is a combination of the outputs of R-Transformer part and Autoregressive part:

$$\hat{Y}_t = m_t + l_t, \tag{10}$$

where $\hat{Y}_t$ denotes the final forecast result of the forecasting module at time stamp $t$.

Overall, the EFA consists of two parts: data preprocessing and forecasting module. (1) The data preprocessing part can standardize the highly dynamic workload of cloud center and use VMD to divide the original sequence into different components (IMFs), which contain the low-frequency and high-frequency information of the original sequence. After using VMD, the interference between different types of information is reduced. (2) The forecasting module composed of R-Transformer and AR model can accurately extract the hidden representations of these components. Among them, R-Transformer uses LocalRNN and the multi-head attention mechanism to obtain the local and global nonlinear relationship of the IMFs, and the AR model is used to obtain the linear relationship of the IMFs. The combination of these components effectively improves the accuracy of forecasting highly dynamic workload in cloud environment.

## V. PERFORMANCE EVALUATION

In this section, we evaluate the performance of the proposed EFA using two real-world cloud center datasets and compare it with RNN-based workload forecasting models and other classic methods [16], [25], [27], [41].

### A. DATASETS

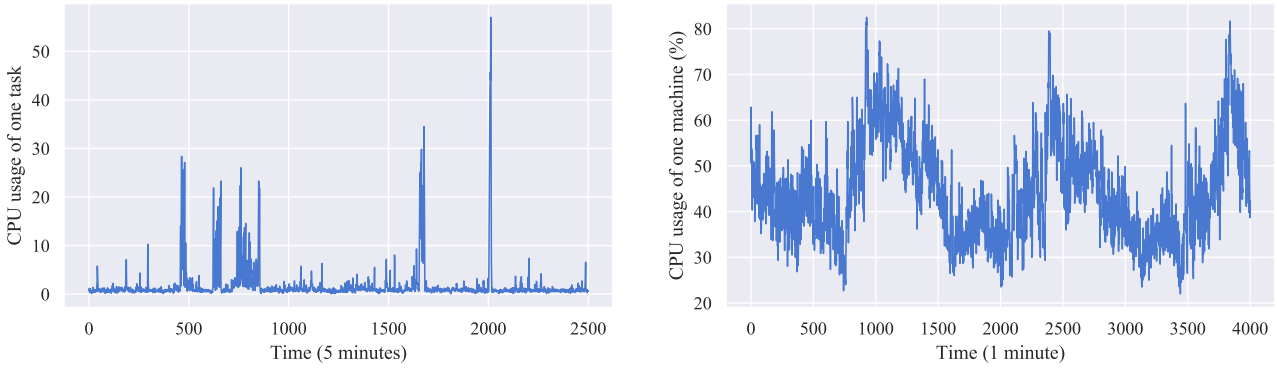
Two real-world datasets are used in the experiments. The first one is Google cluster trace dataset [42], which contains the running information approximately an 12.5k-machine cluster with a span of 29 days during May 2011. The second one is Alibaba cluster trace dataset [6], which includes about 4000 machines with the runtime resource usage in a period of 8 days. In the experiments, we used CPU usage (also known as CPU rate) as the primary performance index of workload.

In addition, to ensure the efficiency and generality of the forecasting methods, two different data types are used for validation. Specifically, for the Google dataset, the target of forecast is the sum of CPU usage of all machines used by a single task. For example, a long-running job with an ID of 6176858948, the job called multiple machines at different times within 29 days. Fig. 4(a) shows the sum of the CPU usage of all these machines at each time point. For the alibaba dataset, we forecast the CPU usage of each individual machine, such as a machine with ID 649 (as shown in Fig. 4(b)). As we can see from Fig. 4, Google workload appears more random and has no obvious periodicity, but its peak is very significant and more difficult to forecast. Alibaba workload displays periodicity, and it varies from 20% to 80% with more frequent changes. Fig. 5 shows the original sequence and its IMFs decomposed by VMD (Alibaba dataset, 3 IMFs).

### B. EXPERIMENTAL SETTING

#### 1) BASELINE METHODS

In order to verify the effectiveness of the EFA, several baseline methods are chosen for comparison as follows:




(a) Google cluster trace dataset

(b) Alibaba cluster trace dataset

**FIGURE 4.** Examples of CPU usage from Google and Alibaba cluster trace datasets. For (a) Google dataset, the CPU usage is the sum of all machines used by one task. For (b) Alibaba dataset, the CPU usage is from one single machine.

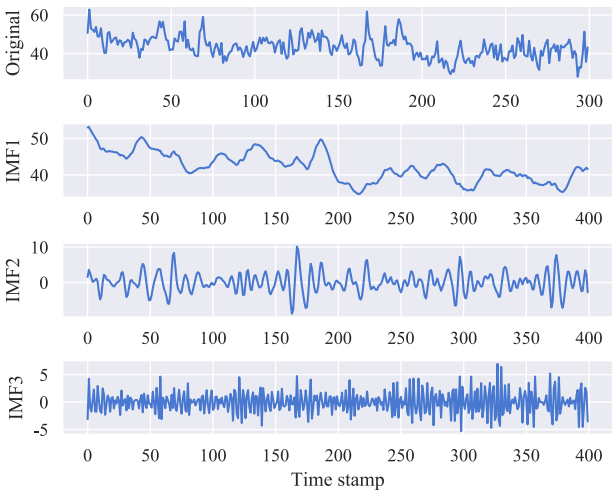



**FIGURE 5.** Use VMD to decompose a sample of the Alibaba dataset. $IMF_1$ can be regarded as a smooth trend change of the original sequence, which belongs to the low frequency part. $IMF_2$ and $IMF_3$ are high-frequency parts that contain small details of the original sequence.

ARIMA [16]: Autoregressive integrated moving average (ARIMA) model is a traditional statistical model for time series forecasting. ARIMA converts non-stationary time series into stationary time series data by using $d$-order difference method. The autocorrelation and partial autocorrelation plots of the historical data are analyzed to determine the order $p$ for the lags of the autoregressive model and the order $q$ for the lags of the moving average model. Finally, ARIMA uses the determined parameters $p$, $d$, and $q$ to predict future workload values.

GRU [27]: The basic gate recurrent unit (GRU) [13] network is a recurrent neural network using GRU cell as the calculation unit. The historical data will be input to the multi-layer GRU network, and the last output value of the last layer will be taken as the forecast value.

LSTM-ED [41]: The long-short term memory (LSTM) encoder-decoder network is similar to the sequence-to-sequence model [43]. LSTM-ED uses LSTM as the basic calculation unit. The historical data is fed into the LSTM encoder network sequentially, and the hidden state and cell state at the end of the encoder are sent to the context vector. The decoder network composed of LSTM outputs the predicted sequence by iteratively decoding the context vector.

LSTM-ED with Attention [25]: Attention based LSTM encoder-decoder network introduces attention mechanism on LSTM-ED. The attention module is part of the decoder network. This module calculates the attention weight of the output of the encoder and the current time step in the decoder. Each weight represents the impact of historical workload on the current workload.

### 2) PARAMETERS SETTING

All methods are implemented in Python 3.6, where the neural network and deep learning methods are implemented using PyTorch 1.3.1 [44]. The experiments are performed on a machine with Intel Core i7-7800X CPU, NVIDIA GeForce RTX 2080 Ti GPU, and 24 GB RAM.

All datasets are divided into training set (80%), validating set (10%) and testing set (10%). The training set is used for forecasting module training, the validating set is used for parameters selection, and the testing set is used to evaluate the performance. Mean square error (MSE) is selected as the loss function during training.

For the proposed EFA, the Adam optimizer is used as the training optimizer and GRU is selected as the calculation unit of LocalRNN. We use grid search to select the length of the history window, the hidden state dimension, the window length of LocalRNN and the number of heads $h$ in the multi-head attention. In detail, search the history window length in {12, 18, 24, 30, 36}, search the hidden state dimension in {64, 128, 256, 512}, search the window length of LocalRNN in {3, 4, 5, 6} and search for the value of $h$ in {4, 8, 12}. In addition, the forecasting length of Google dataset is fixed at 10 minutes, which is 2 time steps while the forecasting length of Alibaba dataset is fixed to 3 minutes, which is 3 time steps. The number of IMFs in both datasets is set to 3. The impact of forecasting length and number of

**TABLE 1. Hyper parameters of the proposed EFA.**

| Hyper Parameter | Value | |
|---|---|---|
| | Google | Alibaba |
| Number of IMFs | 3 | 3 |
| History window length | 24 | 24 |
| Number of training epochs per IMF | 300 | 250 |
| Batch size | 128 | 128 |
| Dimension of hidden state | 256 | 128 |
| Window size of LocalRNN | 3 | 3 |
| $h$ of multi-head attention | 8 | 8 |
| Initial learning rate | 0.001 | 0.001 |

IMFs on performance will be discussed in Section V-D. Some other parameters, such as batch size and learning rate, are determined based on some tuning. The hyper parameters of proposed EFA for Google and Alibaba datasets are shown in Table 1 in detail.

The parameters of the comparison methods are determined based on previous works [16], [25], [27], [41] and some tuning to ensure that each method exhibits its best performance. We perform ten experiments on each method and take the average value as the final result.

#### 3) EVALUATION METRICS

To assess the approach considered in this paper, we used three conventional evaluation metrics defined as follows.

- Mean Squared Error (MSE):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{11}$$

- Root Relative Squared Error (RRSE):

$$\text{RRSE} = \frac{\sqrt{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}}{\sqrt{\sum_{i=1}^{n}(Y_i - mean(Y))^2}} \tag{12}$$

- R-Squared (R2):

$$\text{R2} = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - mean(Y))^2} \tag{13}$$

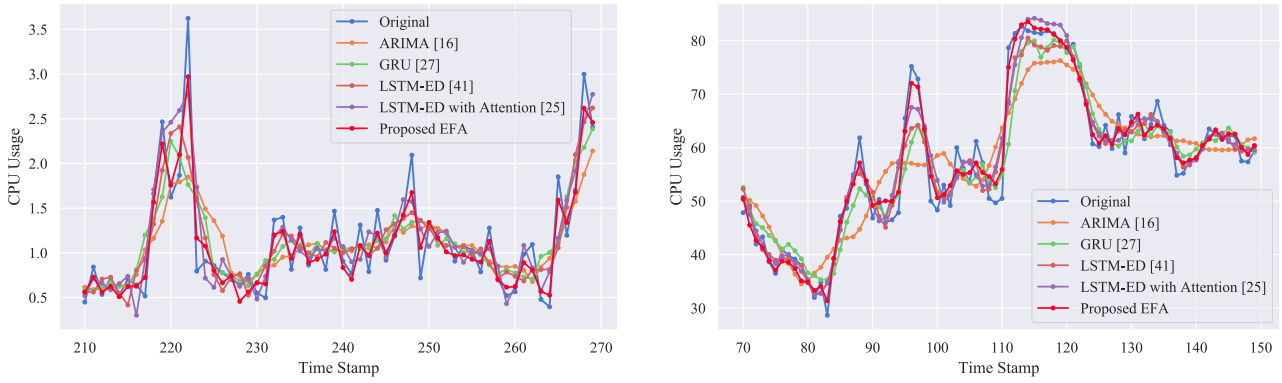
where $Y$, $\hat{Y}$ are original values and forecast values, respectively. The RRSE is a scaled version of the Root Mean Square Error (RMSE) that makes the evaluation results more readable at any data scale [40]. For MSE and RRSE, lower values are better, and for R2, higher value is better.

### C. EVALUATION RESULTS

Table 2 shows the performance of the proposed EFA and baseline methods on Google and Alibaba datasets. Fig. 6 shows the forecasting curves of these methods on the Google and Alibaba datasets. As mentioned in the parameter settings above, for the Google dataset, the default sampling interval is 5 minutes and the forecasting length is 10 minutes. For the Alibaba dataset, the default sampling interval is 1 minute, and the forecasting length is 3 minutes. It can be seen from Table 2 that on the Google and Alibaba datasets, the four deep learning based methods (GRU, LSTM-ED, LSTM-ED with Attention and our proposed EFA) are better than traditional statistical method (ARIMA) in terms of forecasting accuracy. Fig. 6 shows that ARIMA only forecasts future trends and cannot capture sudden changes in cloud workloads. We found that although the ARIMA model is widely used and has a perfect theoretical basis, it is difficult to adapt to the high dynamic load in the cloud environment, resulting in an increase in forecasting error. For deep learning based methods, the performance of the network using encoder-decoder structure and EFA is better than basic GRU. Because the encoder-decoder structure can extract hidden features of the entire sequence. In addition, the attention-based encoder-decoder network (LSTM-ED with attention) can pay attention to the impact of each time step on the forecasting results, enhancing the basic encoder-decoder network. Regarding the proposed EFA, VMD decomposes the sequence into multiple IMFs for independent forecast, reducing the interference between different IMFs. And R-Transformer can not only extract hidden information in a small local window, but also use the multi-head attention mechanism to obtain the

**TABLE 2. The evaluation results on Google and Alibaba dataset.**

| Methods | Google Dataset | | | Alibaba Dataset | | |
|---|---|---|---|---|---|---|
| | MSE | RRSE | R2 | MSE | RRSE | R2 |
| ARIMA [16] | 0.00972 | 0.53359 | 0.73881 | 0.12883 | 0.49762 | 0.72198 |
| GRU [27] | 0.00552 | 0.42004 | 0.78712 | 0.09023 | 0.42012 | 0.80991 |
| LSTM-ED [41] | 0.00419 | 0.36623 | 0.84897 | 0.08112 | 0.37121 | 0.85881 |
| LSTM-ED with Attention [25] | 0.00389 | 0.34023 | 0.88120 | 0.07793 | 0.35281 | 0.88014 |
| Proposed EFA | **0.00338** | **0.31186** | **0.90274** | **0.07121** | **0.32947** | **0.89145** |

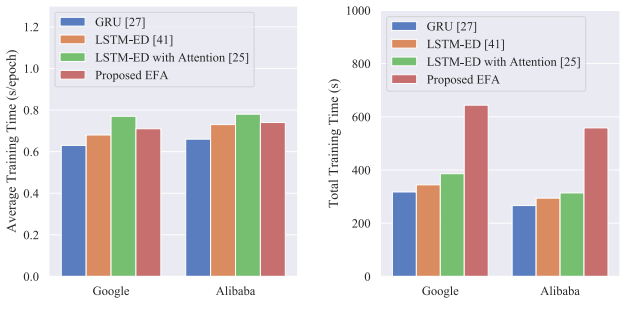



(a) Google dataset (b) Alibaba dataset

**FIGURE 6.** The forecasting performance of the baseline methods and proposed EFA on Google and Alibaba datasets. The EFA can make more accurate forecasting in the face of each time step that changes significantly. This shows that the EFA can adapt to random changes in highly dynamic cloud workloads.

global information of the entire sequence, which significantly improves the accuracy of the forecasting results. More importantly, VMD separates the low-frequency and high-frequency parts of the original sequence. The low-frequency part contains the trend and seasonal characteristics of the sequence, and the high-frequency part contains the details of the sequence. In EFA, the forecasting module trains these IMFs containing different information separately, which can effectively improve the forecasting accuracy. For example, in Fig. 6(a), there was a small drop before the CPU usage surged, and only the EFA predicted this change.

Fig. 7 shows the average training time (each epoch training time) and total training time of each method on the Google and Alibaba datasets. In particular, ARIMA is not like deep learning methods that need to update the weights in an iterations manner (training), so the ARIMA is omitted from the comparison of training time. It can be seen that the fastest training speed of a single epoch is the GRU model, which has a simple structure. The single epoch training of LSTM-ED with Attention model is the slowest. Although the EFA also has an attention mechanism, its LocalRNN part uses a GRU unit with shared weights, which is faster than LSTM, so a single epoch training speed is faster than LSTM-ED with Attention. Since EFA requires separate training for different IMFs, the total training time is slower than the baseline methods. In addition, we found that IMF converges faster than the original sequence during training, so the number of epochs for a single IMF can be less, which has a positive effect on reducing the total training time of EFA.




(a) Average training time (b) Total training time

**FIGURE 7.** Comparison of training time for baseline methods and the proposed EFA on Google and Alibaba dataset. Sub-figure (a) shows the training time of each epoch of these methods, and sub-figure (b) shows the total training time of these methods. For the Google dataset, the number of training epochs of baseline methods is 500. For the Alibaba dataset, the number of training epochs of baseline methods is 400.

### D. DISCUSSION ON THE NUMBER OF IMFs AND FORECASTING LENGTH

In this part, we use VMD to decompose the original workload sequence into $\{0, 2, 3, 4, 5, 6\}$ IMFs respectively, and explore the impact of the number of IMFs on the forecasting results and forecasting efficiency. All experiments used the Google cluster dataset. The model parameters are consistent with Table 1 except for the number of IMFs, and the predicted length is 10 minutes.

Table 3 shows the variance of each IMF and the total training time of the corresponding model under different number of IMFs. The variance of the original workload sequence is 5.075. The variance of each IMF after using VMD is lower than that of the original sequence, which shows that VMD reduces the randomness and volatility of the original sequence. Moreover, the greater the number of IMFs decomposed by VMD, the smaller the mean variance of IMFs, which facilitates the training of deep learning models and achieves more accurate forecasting results. Fig. 8 shows the effect of different number of IMFs on the model forecasting results (measured using MSE). When VMD is not used to process the original data, the forecasting results is the worst. After using VMD, the forecasting accuracy has been significantly improved. And as the number of IMFs increases, the MSE will become smaller. But the cost of improving forecasting accuracy is longer training time, because the greater number of IMFs means training more models.

**TABLE 3. The variance of each IMF and the influence of the number of IMFs on the total training time.**

| Number of IMFs | IMF | Variance | Total training time |
|---|---|---|---|
| 0 | - | 5.075 | 365.22 s |
| 2 | $IMF_1$<br>$IMF_2$ | 3.657<br>0.602 | 461.96 s |
| 3 | $IMF_1$<br>$IMF_2$<br>$IMF_3$ | 3.512<br>0.584<br>0.206 | 643.72 s |
| 4 | $IMF_1$<br>$IMF_2$<br>$IMF_3$<br>$IMF_4$ | 2.455<br>0.971<br>0.440<br>0.181 | 857.16 s |
| 5 | $IMF_1$<br>$IMF_2$<br>$IMF_3$<br>$IMF_4$<br>$IMF_5$ | 2.231<br>1.043<br>0.360<br>0.250<br>0.155 | 1069.10 s |
| 6 | $IMF_1$<br>$IMF_2$<br>$IMF_3$<br>$IMF_4$<br>$IMF_5$<br>$IMF_6$ | 2.182<br>1.049<br>0.323<br>0.254<br>0.112<br>0.116 | 1293.24 s |

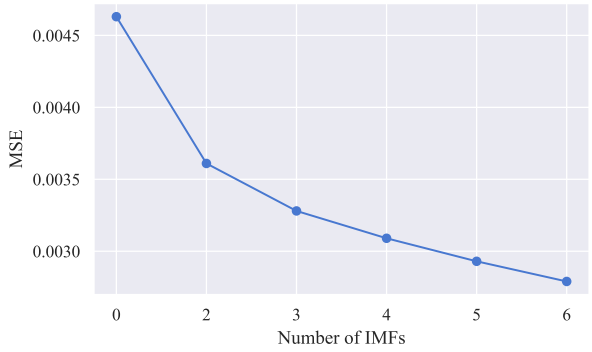



**FIGURE 8. The mean square error of the forecasting results for different number of IMFs. 0 means no VMD is used.**

In order to study the forecasting accuracy at different forecasting lengths, more cases are discussed in Table 4. For the Google dataset, the default sampling time length is 5 minutes, and we resample it to 10 minutes and 30 minutes. Similarly, for the Alibaba dataset, the default sampling time is 1 minute, and we resample it to 5 minutes and 10 minutes. The experimental results were evaluated using mean square error (MSE), and other parameters are the same as in Table 1.

From Table 4, we can see that with the same sampling rate, the model that predicts the longer the future has a higher MSE. This means that the model becomes worse when dealing with longer forecasting intervals. At different sampling rates, overall, the proposed EFA can maintain a low MSE level, which means that if the original sequence is resampled, or using a larger time interval during data collection, the forecasting accuracy can still be guaranteed.

**TABLE 4. Mean square error of different forecasting lengths at different time sampling rate on Google and Alibaba datasets.**

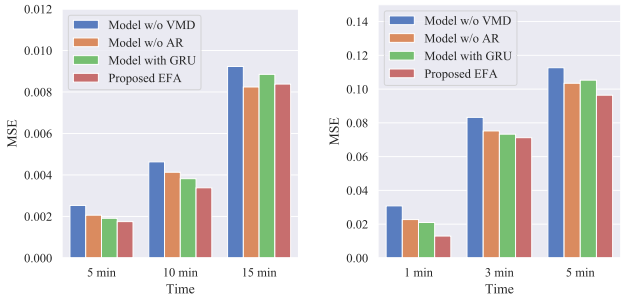
| Dataset | Sample rate | Forecasting length | MSE |
|---|---|---|---|
| Google | 5 min | 5 min | 0.00175 |
| | | 10 min | 0.00338 |
| | | 15 min | 0.00838 |
| | 10 min | 20 min | 0.00334 |
| | | 30 min | 0.00497 |
| | | 40 min | 0.00833 |
| | 30 min | 60 min | 0.00415 |
| | | 90 min | 0.00566 |
| | | 120 min | 0.00861 |
| Alibaba | 1 min | 1 min | 0.01295 |
| | | 3 min | 0.07121 |
| | | 5 min | 0.09637 |
| | 5 min | 10 min | 0.04063 |
| | | 15 min | 0.07727 |
| | | 20 min | 0.09649 |
| | 10 min | 30 min | 0.05362 |
| | | 40 min | 0.07402 |
| | | 50 min | 0.09954 |

### *E. ABLATION STUDY*

To demonstrate the effectiveness of our approach design, we perform an ablation study. Specifically, in order to verify that the improvement in forecast accuracy comes from each component rather than a specific hyper-parameter, we remove each component one at a time in the proposed EFA and compare it with the complete EFA. In addition, we replaced R-Transformer component with GRU to prove the ability of R-Transformer to extract local and global nonlinear information. We name these models as follows:

- *Model w/o VMD*: The model without the VMD component.
- *Model w/o AR*: The model without the Autoregressive component.
- *Model with GRU*: The model without the R-Transformer component and replace it with a basic GRU network.

The ablation study results are shown in Fig. 9. It can be seen that no matter which proposed component is removed




(a) Google dataset (b) Alibaba dataset

**FIGURE 9. The ablation study results on Google and Alibaba datasets.**

from the model, performance will decrease. For example, for the 15-minute forecast length of the google dataset, using GRU instead of R-Transformer raises the MSE from 0.00838 to 0.00885. We have discussed the effect of VMD in Section V-D. Similarly, for the Google dataset, in the case of a forecasting length of 15 minutes, we get worse results with the MSE of 0.00923 without VMD, and the same is true for all results, which shows that VMD can decrease the non-stationarity of highly-dynamic cloud workload and this is beneficial for forecasting module to extract correlations. Overall, the proposed EFA can achieve stable and competitive results on different datasets.

## VI. CONCLUSION

In this paper, we propose a novel Ensemble Forecasting Approach (EFA) for highly-dynamic cloud workload. To reduce the impact of high variance and unstable workload on forecast accuracy, we use VMD to decompose workload into multiple IMFs. To extracted the nonlinear correlation of the input IMF, we further develop a forecasting module composed of R-Transformer, which uses LocalRNN and multi-head attention mechanism to obtain the local and global internal representation of sequences. In addition, we also added an autoregressive model in parallel with R-Transformer to obtain the linear correlation of IMF to improve the robustness of the EFA. Compared with existing forecasting methods, simulation results have demonstrated the effectiveness of proposed approach on real-world datasets from multiple large-scale cloud centers. Finally, we discussed in detail the impact of different numbers of IMFs and the length of the forecasting time on the forecasting results.

One open problem is how to use a non-decomposition and efficient method to reduce the instability and randomness of highly-dynamic workload data, which is a good solution for reducing the training cost of deep neural networks.

## REFERENCES

[1] S. R. Ali, "Cloud computing reliability analysis," in *Next Generation and Advanced Network Reliability Analysis*. Cham, Switzerland: Springer, 2019, pp. 157–187.

[2] A. Ullah, J. Li, Y. Shen, and A. Hussain, "A control theoretical view of cloud elasticity: Taxonomy, survey and challenges," *Cluster Comput.*, vol. 21, no. 4, pp. 1735–1764, Dec. 2018.

[3] M. Masdari and A. Khoshnevis, "A survey and classification of the workload forecasting methods in cloud computing," *Cluster Comput.*, pp. 1–26, Dec. 2019.

[4] M. Amiri and L. Mohammad-Khanli, "Survey on prediction models of applications for resources provisioning in cloud," *J. Netw. Comput. Appl.*, vol. 82, pp. 93–113, Mar. 2017.

[5] I. S. Moreno, P. Garraghan, P. Townend, and J. Xu, "Analysis, modeling and simulation of workload patterns in a large-scale utility cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 2, pp. 208–221, Apr. 2014.

[6] J. Guo, Z. Chang, S. Wang, H. Ding, Y. Feng, L. Mao, and Y. Bao, "Who limits the resource efficiency of my datacenter: An analysis of alibaba datacenter traces," in *Proc. Int. Symp. Qual. Service*, Jun. 2019, p. 39.

[7] M. Alam, K. A. Shakil, and S. Sethi, "Analysis and clustering of workload in Google cluster trace based on resource usage," in *Proc. IEEE Int. Conf. Comput. Sci. Eng. (CSE) IEEE Intl Conf. Embedded Ubiquitous Comput. (EUC) 15th Int. Symp. Distrib. Comput. Appl. for Bus. Eng. (DCABES)*, Aug. 2016, pp. 740–747.

[8] Z. Chen, J. Hu, G. Min, A. Y. Zomaya, and T. El-Ghazawi, "Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 4, pp. 923–934, Apr. 2020.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[10] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 06, no. 2, pp. 107–116, Apr. 1998.

[11] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1310–1318.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*. [Online]. Available: http://arxiv.org/abs/1412.3555

[14] Y. Hu, B. Deng, F. Peng, and D. Wang, "Workload prediction for cloud computing elasticity mechanism," in *Proc. IEEE Int. Conf. Cloud Comput. Big Data Anal. (ICCCBDA)*, Jul. 2016, pp. 244–249.

[15] Z. Amekraz and M. Y. Hadi, "An adaptive workload prediction strategy for non-Gaussian cloud service using ARMA model with higher order statistics," in *Proc. IEEE 11th Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2018, pp. 646–651.

[16] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS," *IEEE Trans. Cloud Comput.*, vol. 3, no. 4, pp. 449–458, Oct. 2015.

[17] P. Saripalli, G. V. R. Kiran, R. R. Shankar, H. Narware, and N. Bindal, "Load prediction and hot spot detection models for autonomic cloud computing," in *Proc. 4th IEEE Int. Conf. Utility Cloud Comput.*, Dec. 2011, pp. 397–402.

[18] J. Guo, J. Wu, J. Na, and B. Zhang, "A type-aware workload prediction strategy for non-stationary cloud service," in *Proc. IEEE 10th Conf. Service-Oriented Comput. Appl. (SOCA)*, Nov. 2017, pp. 98–103.

[19] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," *EURASIP J. Adv. Signal Process.*, vol. 2016, no. 1, p. 67, Dec. 2016.

[20] M. Ghobaei-Arani, S. Jabbehdari, and M. A. Pourmina, "An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach," *Future Gener. Comput. Syst.*, vol. 78, pp. 191–210, Jan. 2018.

[21] G. K. Shyam and S. S. Manvi, "Virtual resource prediction in cloud environment: A Bayesian approach," *J. Netw. Comput. Appl.*, vol. 65, pp. 144–154, Apr. 2016.

[22] P. Singh, P. Gupta, and K. Jyoti, "TASM: Technocrat ARIMA and SVR model for workload prediction of Web applications in cloud," *Cluster Comput.*, vol. 22, no. 2, pp. 619–633, Jun. 2019.

[23] A. A. Rahmanian, M. Ghobaei-Arani, and S. Tofighy, "A learning automata-based ensemble resource usage prediction algorithm for cloud computing environment," *Future Gener. Comput. Syst.*, vol. 79, pp. 54–71, Feb. 2018.

[24] M. Duggan, K. Mason, J. Duggan, E. Howley, and E. Barrett, "Predicting host CPU utilization in cloud computing using recurrent neural networks," in *Proc. 12th Int. Conf. for Internet Technol. Secured Trans. (ICITST)*, Dec. 2017, pp. 67–72.

[25] Y. Zhu, W. Zhang, Y. Chen, and H. Gao, "A novel approach to workload prediction using attention-based LSTM encoder-decoder network in cloud environment," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, no. 1, p. 274, Dec. 2019.

[26] W. Guo, W. Ge, X. Lu, and H. Li, "Short-term load forecasting of virtual machines based on improved neural network," *IEEE Access*, vol. 7, pp. 121037–121045, 2019.

[27] Y. Guo and W. Yao, "Applying gated recurrent units pproaches for workload prediction," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Apr. 2018, pp. 1–6.

[28] Q. Zhang, L. T. Yang, Z. Yan, Z. Chen, and P. Li, "An efficient deep learning model to predict cloud workload for industry informatics," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3170–3178, Jul. 2018.

[29] F. J. Baldan, S. Ramirez-Gallego, C. Bergmeir, F. Herrera, and J. M. Benitez, "A forecasting methodology for workload forecasting in cloud systems," *IEEE Trans. Cloud Comput.*, vol. 6, no. 4, pp. 929–941, Oct. 2018.

[30] K. Dragomiretskiy and D. Zosso, "Variational mode decomposition," *IEEE Trans. Signal Process.*, vol. 62, no. 3, pp. 531–544, Feb. 2014.

[31] R. Salles, K. Belloze, F. Porto, P. H. Gonzalez, and E. Ogasawara, "Non-stationary time series transformation methods: An experimental review," *Knowl.-Based Syst.*, vol. 164, pp. 274–291, Jan. 2019.
[32] A. A. Abdoos, "A new intelligent method based on combination of VMD and ELM for short term wind power forecasting," *Neurocomputing*, vol. 203, pp. 111–120, Aug. 2016.
[33] M. Gendeel, Z. Yuxian, and H. Aoqi, "Performance comparison of ANNs model with VMD for short-term wind speed forecasting," *IET Renew. Power Gener.*, vol. 12, no. 12, pp. 1424–1430, Sep. 2018.
[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[35] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*. [Online]. Available: http://arxiv.org/abs/1810.04805
[36] Z. Wang, Y. Ma, Z. Liu, and J. Tang, "R-transformer: Recurrent neural network enhanced transformer," 2019, *arXiv:1907.05572*. [Online]. Available: http://arxiv.org/abs/1907.05572
[37] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 5244–5254.
[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.
[39] J. Lei Ba, J. Ryan Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*. [Online]. Available: http://arxiv.org/abs/1607.06450
[40] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jun. 2018, pp. 95–104.
[41] H. M. Nguyen, G. Kalra, and D. Kim, "Host load prediction in cloud computing using long short-term memory encoder–decoder," *J. Supercomput.*, vol. 75, no. 11, pp. 7592–7605, Nov. 2019.
[42] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: Format+ schema," Google, Menlo Park, CA, USA, White Paper, 2011, pp. 1–14.
[43] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," 2014, *arXiv:1406.1078*. [Online]. Available: http://arxiv.org/abs/1406.1078
[44] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8024–8035.

**SHAOJING ZHOU** received the bachelor's degree from the College of Civil Engineering and Architecture, Shandong University of Science and Technology, China, in 2018. He is currently pursuing the M.S. degree with the College of Computer Science and Technology, Shanghai University of Electric Power, China. His research interests include cloud computing and deep learning.

**JINGUO LI** (Member, IEEE) received the B.S. degree in information security and the Ph.D. degree in computer science and technology from Hunan University, China, in 2007 and 2014, respectively. He is currently an Associate Professor with the College of Computer Science and Technology, Shanghai University of Electric Power. His research interests include information security and privacy, applied cryptography, and cloud computing.

**KAI ZHANG** received the bachelor's degree in computer science and technology from Shandong Normal University, China, in 2012, and the Ph.D. degree in computer science and technology from East China Normal University, China, in 2017. He visited Nanyang Technological University, in 2017. He is currently an Assistant Professor with the Shanghai University of Electric Power, China. His research interests include data-driven privacy enhanced techniques and cloud security.

**MI WEN** (Member, IEEE) received the Ph.D. degree in computer science from Shanghai Jiao Tong University, Shanghai, China, in 2008. She is currently a Professor with the College of Computer Science and Technology, Shanghai University of Electric Power, Shanghai. Her current research interests include security in wireless sensor networks, cloud security, and privacy preserving in smart grid.

**QIJIE GUAN** received the bachelor's degree in measurement and control technology and instruments from the Jiangsu University of Technology, in 2017. She is currently pursuing the master's degree with the College of Computer Science and Technology, Shanghai University of Electric Power, China. Her research interests include geological modeling and deep learning.