Future Generation Computer Systems 130 (2022) 292–306

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

A multi-output prediction model for physical machine resource usage in cloud data centers

Yongde Zhang [a], Fagui Liu [a,*], Bin Wang [b,*], Weiwei Lin [a], Guoxiang Zhong [a], Minxian Xu [c], Keqin Li [d,e]

[a] School of Computer Science and Engineering, South China University of Technology, GD 510006, China
[b] Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518066, China
[c] Shenzhen Institutes of Advanced Technology Chinese Academy of Sciences, Shenzhen, GD 518000, China
[d] College of Information Science and Engineering and Changsha National Supercomputing Center, Hunan University, Hunan 410082, China
[e] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

ARTICLE INFO

Article history:
Received 3 October 2021
Received in revised form 29 November 2021
Accepted 4 January 2022
Available online 12 January 2022

Keywords:
Cloud data center
Neural network
Noise reduction
Physical machine resource utilization
Time series forecasting

ABSTRACT

Accurate and reliable resource utilization forecasting is critical to achieving efficient resource scheduling in data centers. Traditional prediction methods in cloud computing provide unidimensional output. However, the unidimensional output cannot capture the relationship between multiple dimensions, which results in limited information and inaccurate prediction results. In this paper, we propose CPW-EAMC, a framework that can predict the resource utilization of physical machines in multiple dimensions. This framework consists of two parts: a noise reduction algorithm and a neural network. We propose a noise reduction algorithm CPW to extract data features more precisely and improve the robustness of our prediction algorithm. Then, we establish a multi-dimensional prediction network named EAMC for accurate predictions in multi-steps. Finally, to comprehensively evaluate the model's performance, we propose a novel evaluation standard CMES for model evaluation. Experimental results show that our model has an improvement of 2% to 17% compared with other popular approaches.

1. Introduction

With the accelerated development of information technology in recent years, the construction of cloud data centers is faster than before. Data Center (DC) is an important infrastructure of modern society. According to the statistics [1], about 97% of network traffic is related to DCs. In 2019, the energy consumption of DCs was about 200 TWh, which accounts for about 1% of global electricity consumption, and demand for data services is rising exponentially [2]. The high energy consumption of DCs has become a concern to cloud service providers and governments.

The underloaded hosts in DCs will bring a colossal electricity cost and negatively affect the cloud computing environment [3]. There are many methods to optimize the resource utilization of hosts. Virtual machine (VM) consolidation is one of the schemes applied to migrate VMs into a lesser number of active physical machines (PMs). As a result, the PMs which have no VMs can be turned into a sleep state to save energy [4]. For the VM consolidation algorithm, a key objective is to locate the consolidation's source machines and target machines. Many studies use threshold as the decision variables in their consolidation algorithm [5,6]. They determine the source machines and target machines by comparing the resource utilization at the current moment with the given threshold. However, such methods are difficult to obtain accurate prediction results. At the same time, other studies use machine learning [7] to predict the VM state in the next scheduling interval and determine which VM needs to be migrated. Many prediction models with single output are proposed in cloud computing, which means their predictions tend to be unidimensional. Unidimensional prediction algorithms will restrict our understanding of how the host works. Because a physical machine is a system, each component in one system works together and will affect each other. In unidimensional forecasting, the model can only learn the historical laws of the dimension, but cannot learn the mutual influences between various dimensions. For example, the utilization of hardware like CPU and memory will affect each other. Using a model to predict them separately cannot catch the internal relationship between the key components.

Further, the consolidation algorithms can make more effective scheduling decisions based on more prediction information. And thereby, multi-step forecasting shows its superiority. So, adopting

* Corresponding authors.
E-mail addresses: 201920141144@mail.scut.edu.cn (Y. Zhang), fgliu@scut.edu.cn (F. Liu), wangb02@pcl.ac.cn (B. Wang), linww@scut.edu.cn (W. Lin), cszhongguoxiang111@mail.scut.edu.cn (G. Zhong), mx.xu@siat.ac.cn (M. Xu), lik@newpaltz.edu (K. Li).

https://doi.org/10.1016/j.future.2022.01.002
0167-739X/© 2022 Elsevier B.V. All rights reserved.

a multidimensional prediction model in this scenario is necessary. In this paper, we propose a multi-input–multi-output (MIMO) prediction model with a multi-step-ahead strategy to provide more information for the consolidation algorithm in advance.

As mentioned in [8], the workload of a PM depends on many random factors, both internal and external. These random factors that we call noise will cause some performance fluctuations in a server, harming our analysis. For external factors, for example, the temperature and humidity around the target machine. On the racks of the data center, if the surrounding servers of the target server are under a high load condition, the heat dissipation of these high load machines will increase. These short-term ambient temperature fluctuations will affect the electronic components' performance of the target server. When the performance drops, the utilization of a component for the same task becomes larger. Also, the fluctuation of outside temperature will cause an inaccurate influence on our acquisition hardware. The external temperature fluctuations are one of the noise sources of external factors on the server. For internal factors, if the machine's task arrives with abnormal fluctuations, the utilization data of a recent time will not reflect the machine's actual use, which will prevent our model from catching the fundamental rules. However, the existence of these noises in the original data is short-term, random, and hard to capture. The existing noise reduction methods will bring about excessive smoothing under this scenario so that the original features in the data will be destroyed. Therefore, we need a noise reduction method that can handle this short-term and random noise but retain the characteristics of the data as much as possible.

In response to the previous possible problems, we propose a new PM resource utilization data denoising method and extract the data characteristics in advance. Considering that the noise introduced into the server is generally short-term and small in amplitude. To not affect the subject's data characteristics, we adopt the Complete Ensemble Empirical Mode Decomposition with Adaptive Noise (CEEMDAN) method to divide the original data into multiple Intrinsic Mode Functions (IMFs). Then we use Permutation Entropy (PE) to calculate the noise contained in each IMF. We only smooth the IMFs with a loud noise. At last, we rebuild the data with the processed IMFs. This approach can greatly reduce the loss of essential features caused by the overall smoothing of the original data, and at the same time, it can remove the noise due to minor effects.

After we obtain the processed data, we need a model that can capture the inherent laws of the data. In cloud computing, Artificial Neural Network (ANN) is widely used to predict resource utilization in a system. ANN has the characteristics of flexibility and excellent nonlinear fitting ability. This kind of method can well dig out the patterns hidden in the historical information. For example, when predicting the workload of DCs, many studies based on ANN achieve good results in predicting the workload of servers.

Nevertheless, these studies largely fail to address the issues we mentioned earlier because they ignored the interaction of other factors. Based on such research background, this paper is devoted to studying the ANN-based prediction model. Elman Neural Network (ENN) is a classical network in solving prediction problems and is used widely. The structure of ENN is simpler than many Recurrent Neural Networks (RNNs) like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), which makes it faster in training and less computing resource cost. However, as a kind of recurrent neural network, the single-layer structure of its context layer has a great limitation on its long-term dependence. To go further, the self-feedback coefficient α is limited in the context layer, making it inflexible to adjust its preference to the data of historical information and current information according to the changes of scenario and requirements. Therefore, this paper proposes a novel prediction model based on ENN, which integrates a feed-forward neural network into a recurrent neural network to optimize the long-term dependence and improve the precision in prediction.

This paper aims to build a multi-step forecasting model with multi-input–multi-output (MIMO) based on the historical data collected in cloud data centers. Therefore, we propose an ENN-Attention-MLP-Context with CEEMDAN-PE-Wavelet (CPW-EAMC) model for PM resource utilization forecasting. The main contributions of this paper are as follows:

1. We propose a CPW-EAMC model with MIMO for multi-step prediction. In the scenario with multi-dimensional output, it is tedious and resource-consuming to train a model separately for each dimension of data. Furthermore, this model gathers multi-dimensional outputs into one model and it can preserve the hidden relationship between the dimensions.
2. We propose a noise reduction method CPW for resource utilization data. With this denoising method, the data can be smoothed and its information can be preserved as much as possible so that we can focus on learning the essential characteristics and rules of the data.
3. To overcome the insufficient memory ability of ENN, we propose a novel network EAMC. With attention mechanism, this network enhances feature extraction ability of the current moment and the Multilayer Perceptron (MLP) component strengthens long-term dependency of ENN.
4. We present a comprehensive evaluation metric in space that can avoid the defect of evaluating the performance of a model through a single metric. Simultaneously, it can be carried out when there is no unity among the indicators.

The rest of the paper is organized as follows. Section 2 introduces the related work on noise reduction and time series forecasting technology. Section 3 presents our CPW-EAMC model in detail. Section 4 presents our new evaluation method in this paper. Section 5 evaluates our model and demonstrates our experiment result. Finally, we conclude our work and show our future research directions in Section 6.

2. Related work

As mentioned above, the data collected from the real world contains influences that cause our analysis to be inaccurate. Such influence that brings us negative effects in our analysis is called noise. The multidimensional PM resource utilization data collected from the cloud server belongs to a complex time series. It is necessary for us to reduce the noise of PM resource utilization data before analyzing it.

2.1. Noise reduction algorithm

Traditional noise reduction algorithms include Fourier Transform (FT), Wavelet Transform (WT), Singular Spectrum Analysis (SSA), and Empirical Mode Decomposition (EMD). After Huang et al. proposed EMD in 1998 [9], studies such as Ensemble Empirical Mode Decomposition (EEMD) [10] and CEEMDAN have been proposed as improvements of EMD. Cheng et al. [11] proposed a method that combined EEMD called EEMD-SVD-LWT to reduce the noise of the radar signal. However, the Gaussian white noise added in the decomposition process is not eliminated after finite averaging in EEMD, which leads to residual noise. Therefore, Torres et al. proposed CEEMDAN [12] which can solve the residual noise problem and make the decomposition result more thorough. With the help of CEEMDAN, we can decompose a nonlinear and non-stationary time series signal into multiple Intrinsic Mode Functions (IMFs) and a margin R. Cao et al. built a forecast model of financial series data based on LSTM [13] in which the series data can be preprocessed respectively by the IMFs obtained through CEEMDAN decomposition. Except for EMD and its variant, WT is also a widely used method. WT is the variant of FT which overcomes the limitations of FT in unstable signals. Recent studies combine WT with other algorithms and achieve desirable results. For example, Bento et al. combined WT and bat algorithm in short-term forecasting for power systems [14] and Qiao et al. used a hybrid model based on WT to predict and analyze U.S. electricity prices [15]. What is more, Li et al. [16] proposed a noise reduction algorithm combining CEEMDAN, PE, and wavelet transform for underwater acoustic signal denoising. They divided IMFs from CEEMDAN into noise IMFs, noise-dominant IMFs, and real IMFs. Then, they identified noise IMFs according to mutual information and obtained the noise-dominant IMFs among the remaining IMFs. Once the IMFs are distinguished as noise IMFs, they will be filtered. However, this method is not suitable in our scenario. The noise in our scenario is random, hard to capture, or even slight. If we filter out some of the IMFs, their information will be lost, which is also meaningful to us. Compared with FT, the performance of WT will be better in the context of sudden rise and fall, which makes WT is more suitable in our scenes that fluctuation of our server is generally short-term.

The above research has demonstrated that WT is an effective and animate algorithm. However, WT performs noise reduction processing on the entire signal, and it will also smooth essential parts of the signal, resulting in a lack of important information. Therefore, our CPW divides the original signal into multiple sub-signals and performs noise reduction for those with significant noise. It allows us to preserve as much important information as possible while obtaining good noise reduction performance.

2.2. Time series forecasting technology

After getting the denoised series successfully, we can continue our work to predict and extract the features of the denoised series. We can divide time series forecasting techniques into parametric and non-parametric methods [17]. However, a parametric method is unfriendly to many researchers as it requires researchers to be proficient in the computational mathematics of their business field. Furthermore, the parametric method shows its limitations when facing a complex and changeable time series. In non-parametric methods, machine learning is an exemplary method. As an essential part of machine learning, ANNs have gained much attention from scientists. ANN has an excellent non-linear fitting ability, and it can extract features from data through training. For example, the Ref. [18] uses MLP in exchange rate prediction. As a feed-forward neural network, MLP lacks the ability of memory and causes gradient explosion or gradient disappearance. As a result, a plain feed-forward neural network has disadvantages when solving a long-term forecasting problem.

Recurrent Neural Network (RNN) can overcome the disadvantages of feed-forward neural network [19]. RNN is widely used in time series prediction and natural language processing due to its memory ability of historical data in its network structure. For example, Hu et al. [20] used particle swarm optimization (PSO) and gradient descent (GD) for aggregation and combined LSTM for trend following while Yang et al. [21] used LSTM with feature enhancement for traffic flow prediction. Moreover, LSTM can deal with non-uniform data. The Ref. [22] processed non-uniformly sampled data with LSTM. Although LSTM is powerful, standard LSTM cannot fully capture all the different effects on target series in multivariate time series prediction tasks. Therefore, Hu et al. [23] used TG-LSTM network for multivariate time series prediction. More and more studies have been trying to combine their algorithm with an attention mechanism in recent years. For example, Lin et al. [24] used LSTM with attention mechanism in electricity consumption forecasting. However, LSTM consumes many resources during training and testing. So, a network unit called GRU is proposed. GRU can achieve similar performance as LSTM but requires fewer resources. Niu et al. [25] improved GRU based on attention mechanism and used it in wind power forecasting. Although GRU needs fewer resources than LSTM, both of them still face the problem of large resource consumption and long training time. A network called Elman Neural Network (ENN) can be trained faster than those as it has fewer parameters in its structure. Research like [26] has shown ENN's ability to feature extraction and the Ref. [27] had shown that ENN could do well in series processing. For instance, Zhang et al. [28] improve ENN with piecewise weighted gradient for time series prediction. Also, ENN is used to solve the energy consumption problem of data centers. Wu et al. [29] built a power consumption model of cloud servers with native ENN. And a modified ENN is used for atrial fibrillation signals classification in [26]. ENN has been proved by many studies and experiments to have outstanding performance in time series prediction.

However, ENN's memory ability is constrained by the single-layer neuron of its context layer, leading to unstable prediction performance and accuracy. To learn the rules of PM resource usage utilization and predict it, we need to combine as much historical information as possible to obtain historical rules. The resource utilization of the server is affected by the user's rules and the logic of the scheduling algorithm. Therefore, we need long-term dependence of the network to capture the contextual correlation of utilization. Our model presented in this paper uses a component to surmount the shortcoming of the original structure. In addition, we enhance its ability to fit periods with significant changes with an attention mechanism.

2.3. Multi-dimensional forecasting technology

In the prediction of time series, many models have unidimensional output even though their input is multi-dimensional. The overall energy consumption of one PM is closely related to different hardware components. It is necessary to build a forecasting model with multi-dimensional output of PMs. Bao et al. [30] used multi-step Support Vector Machine (SVM) in time series prediction. However, they must apply a MIMO strategy with their model, or their model would degenerate into H dispersed models (H is the dimension of input data). Some researchers notice the significance of the MIMO model to the real world. Zhou et al. [31] developed a deep multi-output LSTM neural network (DM-LSTM) for air quality forecasting. In the real world, most of the problems are caused by multiple factors. When solving time series forecasting problems, it is necessary to consider that they are affected by multi-dimensional factors. And as a result, it is significant to study the time series forecasting problem with multiple dimensions in cloud computing.

2.4. Resource usage prediction in cloud computing

With the development of forecasting technology, more and more researchers apply forecasting techniques to cloud computing. In [5], it uses Extreme Learning Machine for CPU utilization prediction of each PM. However, a simple network for forecasting cannot satisfy the accuracy. Therefore, Sima et al. [32] propose a Wavelet-GMDH-ELM model for workload prediction that contains CPU, storage, and network resources. The Wavelet of the model is used for data analysis and noise reduction, while the rest is for workload prediction. This model uses Wavelet for noise reduction to improve the prediction accuracy.

To improve the forecasting accuracy, Kim et al. [33] propose a Sequence-to-Sequence-LSTM (Seq2Seq-LSTM) based on the robust framework called Sequence-to-Sequence in time series prediction for energy consumption prediction. Moreover, Hoang et al. [34] use LSTM-Encoder–Decoder for host load prediction. Time series forecasting technology is a hot and potential method in cloud computing. Therefore, we obtain a prediction framework with good robustness and high accuracy by combining an improved noise reduction method with a novel network.

Therefore, in this paper, we propose a CPW-EAMC framework to process the data in the cloud cluster in the early stage and provide accurate multi-step multi-dimensional forecasts for the resource utilization of physical machines.

3. The proposed method

3.1. Motivation

In a cloud data center, the submitted tasks mainly include CPU-intensive tasks, memory-intensive tasks, and disk I/O-intensive tasks. When analyzing the performance of PMs, the scheduling algorithm in a DC has a greater impact on the resource utilization of one single PM. Different scheduling strategies will execute different scheduling schemes for the submitted tasks, resulting in workload differences between PMs. Predicting the resource utilization of PMs can provide more information to the scheduling algorithm for effective and accurate decisions, which can make the workload of a PM rational. In addition, within our knowledge, most research has ignored the implicit mutual relationship between hardware. With MIMO, we can capture the hidden relationship between hardware in one model instead of observing them individually.

In this scenario, we found that the prediction accuracy of ENN has a good performance with a short training time. However, ENN has a critical shortcoming in this scenario that is unstable. Therefore, it would be encouraging if we could improve ENN to make its prediction more stable.

For accurate prediction and stable performance, we propose CPW-EAMC. With the help of the noise reduction algorithm CPW, we can enhance the robustness of models and optimize the generalization ability and the fitting ability of the model.

3.2. CEEMDAN-PE-Wavelet (CPW)

CEEMDAN is an algorithm that decomposes a complete signal into multiple IMFs. CEEMDAN is an improved algorithm of EMD. CEEMDAN adds adaptive white noise to each stage of its decomposition, which can eliminate model aliasing and reduce the reconstruction error to the minimum.

PE is an indicator used to measure the complexity of time series [35]. PE is to add a sorting step when calculating the complexity between reconstructed subsequences. Suppose we have obtained a time series of length $L+1$, $Ts = \{ts_i | i = 0, 1, 2, \dots, L\}$ where ts_i is the value at moment i. We need to reconstruct, sort, and calculate the permutation entropy of the original sequence Ts. The greater permutation entropy, the more complex the time series.

The resource utilization of physical machines is mainly determined by the submitted tasks. However, the time when tasks are submitted by users and the type of tasks are completely random. It is one of the main reasons for the unstable use of physical machine resources. FT is a classical method in time series processing. Nonetheless, FT has obvious defects in managing non-stationary time series. Luckily, as an improvement of FT, WT overcomes the limitations of FT in unstable series processing. Therefore, WT is suitable for tackling the problems in this paper.

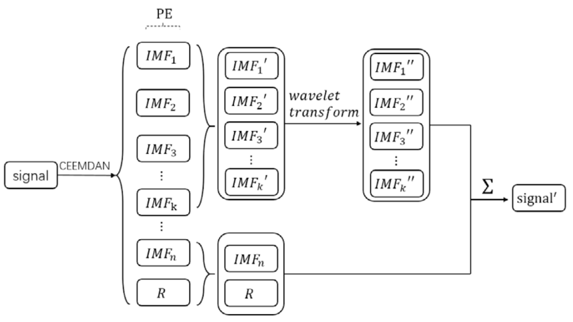


Fig. 1. Process of CEEMDAN-PE-Wavelet.

We propose a noise reduction algorithm called CEEMDAN-PE-Wavelet (CPW), which can be shown in Fig. 1. This algorithm decomposes the original time series through CEEMDAN into several IMFs and a margin R. The margin R is the residual amount generated after the decomposition of IMFs. We can regard each IMF as new time-series data. For the multiple time series data we newly obtained, we calculate permutation entropy for each IMF. We need to define a threshold *tk* of permutation entropy. We calculate the permutation entropy for each IMF decomposed from the original signal and sort IMFs based on the descending order of permutation entropy. We can aim at the high noise subsequence for denoising through this method and avoid the disadvantage of reducing the signal containing important information while weakening the noise.

We select the IMFs in which permutation entropy is higher than the given threshold *tk* and group them into a set IMF' where $IMF' = \{IMF_1', IMF_2', \dots, IMF_k'\}$. Then, we denoise each $IMF_i' \in IMF', i \in [1, k], i \in N$ and get the collection of processed IMFs, $IMF'' = \{IMF_1'', IMF_2'', \dots, IMF_k''\}$. Finally, we add up all the IMFs and the margin R we have including the IMFs whose permutation entropy is lower than the given *tk* to rebuild the denoised signal. This algorithm is described as Algorithm 1. The $WT()$ function in line 16 in Algorithm 1 is the wavelet transform function. Since each dimension has its own characteristics, we execute the CPW algorithm separately on each dimension.

3.3. ENN-Attention-MLP-Context (EAMC)

As is shown in Fig. 2, the traditional ENN uses a layer of neurons as a context layer. The orange lines in Fig. 2 show the trace of state from the hidden layer to the context layer. α is a self-feedback coefficient, which is to merge the previous context state with the state at that moment. The context layer's primary function is to act like the recurrent component of RNN for history information recording. However, only one layer of neurons constrains ENN's ability to memorize history information.

The application of attention mechanisms in the field of artificial intelligence is relatively extensive. The attention mechanism mainly imitates human beings' behavior to focus on certain important areas when observing the image. There are many changes in the attention mechanism. Soft attention and hard attention are typical representatives. The attention mechanism mainly gives the value in a vector or a matrix more weight to the focus area. The value of the hard attention is in $\{0, 1\}$, while the value of soft attention is in a range of $[0, 1]$. The main problem of the hard attention mechanism is that its value is either 0 or 1. It will cause a large amount of information loss if the hard attention mechanism is applied to continuous data. However, soft attention

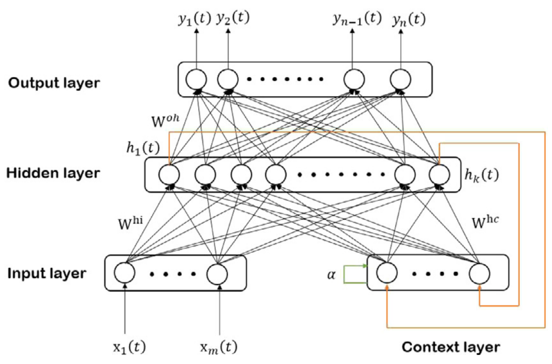


Fig. 2. Structure of ENN.

can avoid this problem, for it can maintain the continuity of data while highlighting the key areas in the continuous data by adjusting the weight in the range of [0,1].

Algorithm 1 CEEMDAN-PE-Wavelet.

```
Input:
    The signal to be denoised, $signal$
    The embedding dim of the permutation entropy, $em$
    The delay time of the permutation entropy, $dt$
    The permutation entropy threshold of the IMF need to be denoise, $th_{pe}$
    The threshold used in wavelet transform, $th_{wa}$
Output: The signal denoised by the algorithm, $signal'$
 1: Initialize PEs with blank list, $PE_{sort}$ with blank dictionary
 2: $IMFs \leftarrow CEEMDAN(signal)$
 3: for $each\_IMF$ in $IMFs$ do:
 4:     PE $\leftarrow$ permutation_entropy($each\_IMF$, $em$, $dt$)
 5:     $PE_{sort}$ $\leftarrow$ {PE, each_IMF}
 6: end for
 7: $PE_{sort} \leftarrow sort(PE_{sort})$
 8: for $each\_PE$ in $PE_{sort}$ do
 9:     if $each\_PE > th_{pe}$ then:
10:         $IMF \leftarrow PE\_sort[each\_PE]$
11:         $PE_{sort}[each\_PE] \leftarrow WT(IMF, th_{wa})$
12:     else
13:         $IMFs$ $\leftarrow$ all IMF by sorted order from $PE_{sort}$
14:     end if
15: end for
16: $signal' \leftarrow \sum IMFs$
17: return $signal'$
```

As we mentioned before, MLP is widely used in many areas, including time series forecasting. The Ref. [36] is an example of time series forecasting with MLP. We can know that MLP has its ability in feature extraction. The MLP (shown in Fig. 3) involves an input layer, an output layer, and at least one hidden layer between the input layer and the output layer [37]. The information in MLP is transmitted forward through layers of neurons. The dimensions of input and output data determine the number of neurons in the input and output layer of the MLP.

With this helpful tool, we proposed a new network based on ENN called ENN-Attention-MLP-Context (as shown in Fig. 4) to enhance the memory ability of ENN. First, we use an MLP to replace the context layer of ENN. Using MLP to expand the context layer can improve time dependence enhancement, making the affected time range larger. Nevertheless, by combining the feature extraction capabilities of MLP, it can extract the features from historical data and store the features inside the subnet

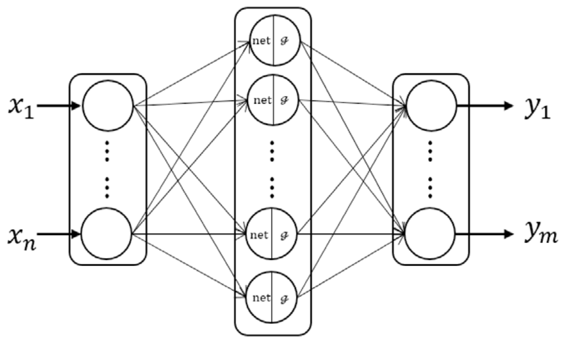


Fig. 3. Structure of MLP.

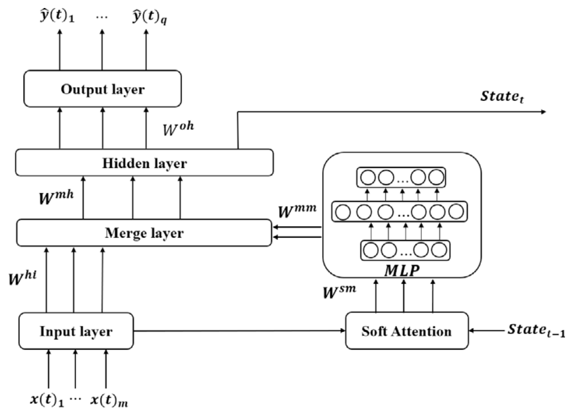


Fig. 4. Structure of ENN-Attention-MLP-Context.

of MLP. This expansion is conducive to enhance the long-term dependence of ENN and the capability to fit time series with internal hidden features.

For the sample at time t, $X(t) = \{x(t), y(t)\}$, $x(t) \in \mathbb{R}^m$, $y(t) \in \mathbb{R}^q$. We assume that the weight between input layer and output layer is W^{hi}. The weight between attention layer and MLP is W^{ms}. MLP has three layers of neurons. The weight between hidden layer and MLP is W^{hm}, the weight between output layer and hidden layer is W^{oh}, and W^{mh} is the weight between merge layer and hidden layer while W^{mm} is the weight between the output of MLP and the merge layer. In input layer, the dimensions of input data and the number of neurons in hidden layer are often inconsistent. We need to map the input data to the same dimension as the hidden layer. Among the weights listed above, $W^{hi} \in \mathbb{R}^{m\times i}$, $W^{ms} \in \mathbb{R}^{n\times n}$, $W^{hm} \in \mathbb{R}^{n\times n}$, $W^{oh} \in \mathbb{R}^{q\times n}$, $W^{mh} \in \mathbb{R}^{m\times h}$, $W^{mm} \in \mathbb{R}^{n\times m}$. We define the self-feedback coefficient between the output of MLP and the input from input layer as α. The number of hidden layers is H. The relationship of each layer can be expressed as follows:

Input Layer:

$$x_i(t) = \text{lin}(x(t)), x_i(t) \in \mathbb{R}^n \tag{1}$$

Merge Layer:

$$m(t) = (1-\alpha) \cdot W^{hi}(t) \cdot x_i(t) + \alpha \cdot W^{mm}(t) \cdot x_m(t) \tag{2}$$

Hidden Layer:

$$h(t) = m(t) \cdot W^{mh}(t) \tag{3}$$

Inside Attention Layer:

$$x_{att\_in}(t) = [x_i(t), State_{t-1}] \tag{4}$$

$$x_{att\_out}(t) = W^{s-o}(t) \cdot x_{att\_in}(t), W^{s-o} \in \mathbb{R}^{(n+n)\times n} \tag{5}$$

Output Layer:

$$\mathrm{net}^{o}(t) = W^{oh}(t)h(t) \tag{6}$$

$$\hat{y}(\mathrm{t}) = \mathrm{g}\left(\mathrm{net}^{o}(t)\right) \tag{7}$$

We can get the calculation expression of the MLP in the context module through the description of MLP:

$$MLP_{in}(t) = W^{sm}x_{att\_out}(t) \tag{8}$$

$$\mathrm{net}\,h_j(t) = w_{j-1}\,\mathrm{net}\,h_{j-1}(t) + b,\ 1 \le \mathrm{j} \le H \tag{9}$$

$$h_j = f\left(\mathrm{net}\,h_j(t)\right) \tag{10}$$

$$x_m(t) = MLP_{\mathrm{out}}(t) = W^{hm}h_H(t) \tag{11}$$

$x_i(t)$ in (1) is the linear mapping result of input and lin(Δ) is the linear mapping function. $x_m(t)$ in (2) is the output of MLP and α is the self-feedback coefficient. Through adjusting α, we can adjust the ratio of historical information to the influence of the input information at the current moment. And we can decide whether the network will be more affected by current information or historical information so that our network can adapt flexibly according to the changes in our scene. $State_{t-1}$ in (4) is the state from hidden layer of the last moment $t-1$ and $x_{att\_out}(t)$ in (5) is the output of Attention Layer. $\hat{y}$(t) in (7) is the prediction result of the network. In (8), $MLP_{in}(t)$ is the input of MLP at time t and net $h_j(t)$ in (9) is the state of hidden layer in MLP where $1 \le j \le H, j \in N$. h_j in (10) is the output of jth hidden layer and $f(\Delta)$ is activation function. In our network, the activation function in MLP is ReLU while the activation function in output layer is sigmoid. $MLP_{out}(t)$ in (11) refers to the output of MLP at time t. h_H (t) in (11) is the output of the last layer of hidden layer.

Unlike traditional ENN, we merge the input at the current moment through the attention mechanism and the state of the context layer. This method can increase the impact of the current data on historical information. During network training, we use MSELoss (12) as the loss function.

$$loss\left(\hat{y}_i, y_i\right) = \left(\hat{y}_i - y_i\right)^2 \tag{12}$$

To prevent serious overfitting and enhance the robustness of our network, we use L2 regularization on the loss function. The objective function we need to optimize can be expressed according to the following formula:

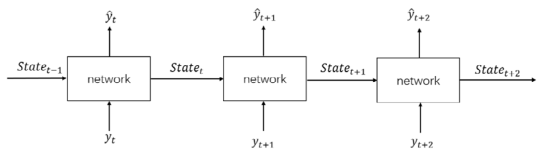
$$\mathrm{E(t)} = \frac{1}{2}(\hat{y}(t) - y(t))^{\top}(\hat{y}(t) - y(t)) + \frac{C}{2}W^{\top}(t)W(t) \tag{13}$$

In the formula, $\hat{y}(t)$ is the predicted value of our network and $y(t)$ is the true value of the predicted value at that moment. $W(t)$ is the weight of the entire network at time t. $W \in \left\{W^{hi}, W^{ms}, W^{hm}, W^{oh}, W^{s-i}, W^{s-o}\right\}^{\top}$. C is the regularization coefficient.

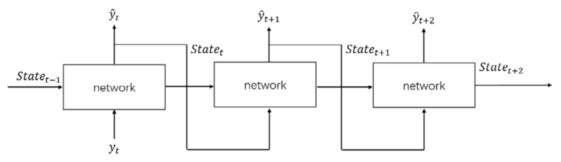
3.4. Prediction strategy

This article divides the prediction strategy into the training phase and prediction phase. In the training phase, our primary purpose is to learn the trends from historical data. In order to enable the network to fit the historical data during training better, we use the data in label as teacher signal to guide the training of the model. Therefore, we use the actual value as a label in our training strategy in Fig. 5a.

However, unlike the training phase, the prediction phase mainly tests the accuracy of the model prediction. Therefore, we adopt the method of cyclic prediction. Our prediction for a time period is based on the previous moment's prediction result, which is more in line with the actual forecasting process. The strategy we use in prediction phase is shown in Fig. 5b.



(a) Strategy of Training



(b) Strategy of Testing

Fig. 5. Strategy of training and testing.

At time t, the output of $\hat{y}(t)$ consists of q dimensions(as shown in Fig. 4). These dimensions are the predicted values of each hardware indicator we want to predict. Then, we obtain our multi-step prediction according to the step size we want in advance. The strategy for obtaining multi-step forecasts is as shown in Fig. 5.

4. Improved evaluate metric

The commonly used evaluation criteria in the research of time series forecasting are MAE, RMSE, and MAPE. The model established in this paper is a multi-output model; as a result, we need to adjust the original calculation methods of MAE, RMSE, and MAPE to a certain extent. This article uses the strategy of averaging the output data when calculating the multi-dimensional output. The formula for calculating our adjusted error is expressed as (14), (15), (16).

$$\mathrm{MAE(X, h)} = \frac{\sum_{j=1}^{dim}\frac{1}{m}\sum_{i=1}^{m}\left|\widehat{y}_{ji} - y_{ji}\right|}{dim} \tag{14}$$

$$\mathrm{RMSE(X, h)} = \sqrt{\frac{\sum_{j=1}^{dim}\sum_{i=1}^{m}\frac{1}{m}\left(\widehat{y}_{ji} - y_{ji}\right)^2}{dim}} \tag{15}$$

$$\mathrm{MAPE} = \frac{\sum_{j=1}^{dim}\frac{1}{m}\sum_{i=1}^{n}\left|\frac{\widehat{y}_{ji}-y_{ji}}{y_{ji}}\right|}{dim} \tag{16}$$

The dim in (14), (15), (16) is the dimension of output data and $\widehat{y}_{ji}$ is the ith prediction output of jth dimension. y_{ji} is the true value of ith datapoint of jth dimension and m is the length of data. MAE, RMSE, and MAPE are indicators for judging the accuracy of models. In many cases, the evaluation criteria of the model rarely have a consistent optimal situation. Therefore, how to comprehensively evaluate multiple evaluation criteria is a problem worthy of study. For example, in our three indicators, RMSE is greatly affected by outliers. If encounters a few data points with large deviations, the calculation result of RMSE will be large, even if the overall fitting effect is good. A relatively simple way to solve this problem is to comprehensively calculate these three criteria by adding a summation method which is shown in (17). We need to set three weights α, β, and γ. However, this calculation method (17) has the main problem, that is, how to set an appropriate weight. The results of different weights will make the final calculation result have an increased deviation. So, it is too subjective for the researcher to set this weight subjectively

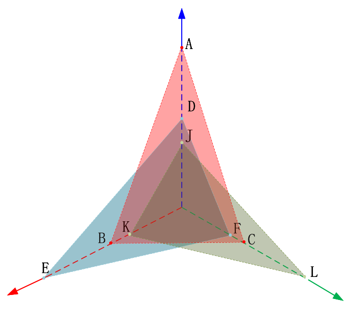


Fig. 6. The base area of different space tetrahedrons with different colors.

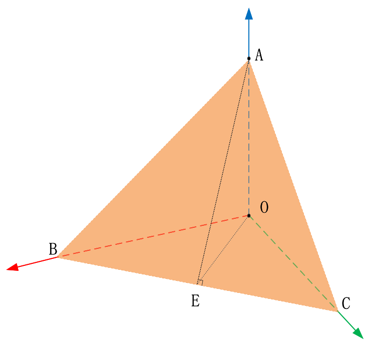


Fig. 7. We start from point A and draw a perpendicular line to BC at point E. Therefore, $AE \perp BC$.

to prefer a certain criterion perceptually.

$$S = \alpha \cdot MAE + \beta \cdot RMSE + \gamma \cdot MAPE \tag{17}$$

In order to solve the problem that these three evaluation standards in the comparison model are difficult to be consistent and optimal, we propose a cartesian coordinate based multi-standard performance measurement evaluation standard (CMES), a comprehensive evaluation standard based on space area.

We regard MAE, RMSE, and MAPE as the three axes in the cartesian coordinate system. The area we calculate is the base area of space tetrahedron, as shown in Fig. 6.

In this paper, we use three criteria as the three axes in the spatial rectangular coordinate system. Here, we randomly assume that MAE is a', RMSE is b', and MAPE is c'. Then we can use (18) calculate the length of the hypotenuse:

$$l_1 = \sqrt{a'^2 + b'^2}, l_2 = \sqrt{b'^2 + c'^2}, l_3 = \sqrt{a'^2 + c'^2} \tag{18}$$

After calculating the lengths of these three hypotenuses, we can calculate base area of tetrahedron with (19) and (20):

$$p = \frac{l_1 + l_2 + l_3}{2} \tag{19}$$

$$\text{CMES} = \sqrt{p\,(p - l_1)\,(p - l_2)\,(p - l_3)} \tag{20}$$

To illustrate the feasibility of this method as a mapping of three error standards. We will prove that the base area of the tetrahedron is positively correlated with the length of three axes. For we need the base area of the tetrahedron, we only focus on the tetrahedron $OABC$ in the space, as shown in Fig. 7. Here, we assume that $OA = a$, $OB = b$, $OC = c$. It is clear that $a, b, c \geq 0$

We start from point A, with O as the origin, draw a perpendicular line to BC at point E. Then, $AE \perp BC$ and $S_{\triangle ABC} = \frac{BC \cdot AE}{2}$. Since A, B, and C are on the three axes of space rectangular coordinate system respectively, $AO \perp \Delta BOC$. As a result, $AO \perp BC$. Because $AE \cap AO = A$, we can know that $BC \perp \Delta AOE$. Therefore, $OE \perp BC$.

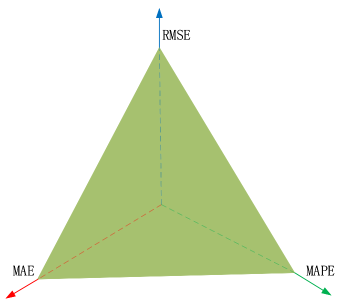


Fig. 8. Result of proposed evaluated method. MAE, RMSE, MAPE are the three axes in the cartesian coordinate system. The bottom area of the space tetrahedron is the comprehensive evaluation.

Obviously, we can get $OE = \frac{b \cdot c}{\sqrt{b^2+c^2}}$, because $S_{\triangle BOC} = \frac{OE \cdot BC}{2} = \frac{BO \cdot OC}{2} = \frac{b \cdot c}{2}$. With $AE = \sqrt{AO^2 + OE^2} = \sqrt{\frac{b^2 \cdot c^2}{b^2+c^2} + a^2}$, we can get $S_{\triangle ABC} = \frac{BC \cdot AE}{2} = \frac{\sqrt{b^2+c^2} \cdot \sqrt{\frac{b^2 \cdot c^2}{b^2+c^2} + a^2}}{2}$.

Then calculate the partial derivative of $S_{\triangle ABC}$ to a. We can get $\frac{\partial S_{\triangle ABC}}{\partial a} > 0$. Without loss of generality, if we use the above proof to draw a vertical line starting from the vertices B and C, we can still get the result that $\frac{\partial S_{\triangle ABC}}{\partial b} > 0$, $\frac{\partial S_{\triangle ABC}}{\partial c} > 0$. The result of partial derivative shows that the area of $S_{\triangle ABC}$ increases monotonically with a, b, and c. Therefore, the larger calculation result of $S_{\triangle ABC}$, the greater overall error; the larger area of $S_{\triangle ABC}$, the lower performance of the model. Our calculation of the comprehensive evaluation of the error can be described in Fig. 8.

If the value of a certain criterion is much larger than the other two criteria, we can also use the square root result of this dimension to maintain the balance of the comprehensive result. By mapping the three axes to space, the linear relationship between area and axis in-plane described in [17] can be removed while treating the three criteria equally.

To enable the three evaluations to be better integrated, we hope that the various dimensions in the expression should not be too direct. By calculating the base area of the tetrahedron, we use Eq. (18) to increase the correlation between these three dimensions. Through this approach, we have eliminated the simple correlation between the three dimensions in traditional calculation methods.

5. Experimental results

In this section, we will examine our model's performance with the data collected in an actual cloud environment. We conducted experiments on our noise reduction algorithm and our overall prediction model. In addition, ablation experiments prove that our improvements to the model are effective and necessary. To verify the usability and robustness of our model under different architectures, we selected a dataset collected from our cluster based on ARM architecture for the experiment. The specific description of each dataset is described in subsections A and B.

5.1. Dataset A: Alibaba Cluster Trace

In this part, we will introduce our experiment based on Alibaba Cluster Trace.[1] The Alibaba Cluster Trace Program is published by Alibaba Group [38]. This program contains cluster-trace-v2017 and cluster-trace-v2018. Our experiment uses cluster-trace-v2018 dataset whose sampling interval is 10 s. However,

[1] https://github.com/alibaba/clusterdata.

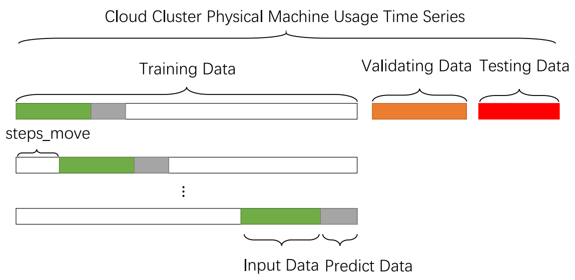


Fig. 9. Data consolidation strategy.

the sampling interval of cluster-trace-v2017 is 300 s in 12 h. As a result, cluster-trace-v2017 has too little data to train a model since it only contains 145 data points for one PM.

Multiple data tables are provided in the dataset. There are seven dimensions of data in the table that provides physical machine performance data includes CPU utilization, memory utilization, men_gpsb(normalized memory bandwidth), net_in (normalized incoming network traffic), net_out (normalized outgoing network traffic), and disk I/O ([0, 100], abnormal values are of -1 or 101). Each PM has its csv document in this dataset, which is named by its machine ID to record its resource utilization. However, in the raw data, the amount of missing data in mem_gps and mkpi is relatively large, resulting in less reference for using these two dimensions. Therefore, the dimensions we used in this experiment were CPU utilization, memory utilization, net_in, net_out, and disk I/O.

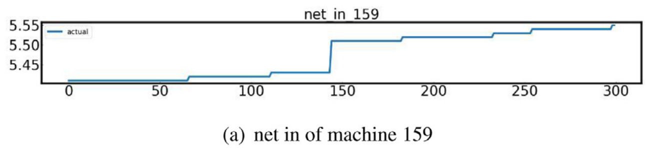
5.2. Dataset B: ARM-based Cloud Computing Cluster

In the ARM-based Cloud Computing Cluster, we collect PM resource utilization from our cloud computing cluster. Our cluster is made up of five servers based on ARM architecture. The hardware configuration of these five servers is the same. Each server has 2 Kunpeng 920 CPUs and each CPU contains 48 cores. And the memory of a server is 256G. The operating system is CentOS7. We use Openstack as a management platform for our cluster. We use stress-ng to add load to the cluster. The load we added includes CPU-intensive tasks, memory-intensive tasks, disk I/O-intensive tasks, and net-intensive tasks. Our sampling interval is 5 s. We used the collected 10,000 data points for experiments. What we collected in the cluster is mainly the utilization of CPU. The CPU utilization we collected is divided into two dimensions: cpu_user and cpu_system. Our data collection software does not directly collect the utilization of resources other than the CPU. In addition, the IO operation simulated by stress-ng will often become an operation on the memory under the influences of the operating system, which will lead to the inaccurate utilization of memory and disk I/O. Therefore, the data we mainly use in our experiment is the dimension of CPU.

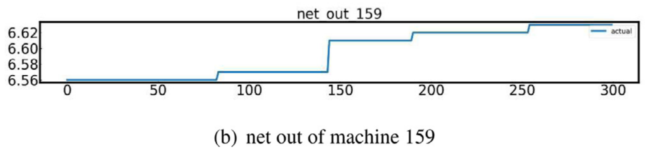
5.3. Data preprocessing strategy

The dataset used for time series forecasting is often a continuous time series with a large amount of data. It is irrational for us to train our network with the entire dataset. Therefore, we need to divide our data into several groups. Our strategy for separating data is shown in Fig. 9.

We divide the entire dataset into a training set, a validation set, and a test set in chronological order. The test set is used to test the model's performance, while the validation set is used to evaluate the performance during a stage of training. The training set is used to train the model. Fig. 9 shows our organization of the



(a) net in of machine 159



(b) net out of machine 159

Fig. 10. Network usage of Machine 159.

data more intuitively. The proportion of training set, validation set, and test set in our experiment is about 0.8, 0.1, 0.1. In the training set, we set the size of the time window as 20 since this parameter is widely used in many studies. After obtaining the label of predicted data according to the prediction step, the starting point of the next group will move back *step_move* steps.

5.4. Denoise algorithm experiment

To illustrate the superiority of the noise reduction algorithm used in this article, we evaluated the noise reduction algorithm under the Alibaba Cluster Trace dataset. We use SNR and RMSE to evaluate our noise reduction algorithm. SNR is signal to noise ratio, which is described in (21). SNR is used to measure the ratio of useful components to noise in the time series before and after noise reduction. The larger SNR, the better effect.

$$\begin{aligned} \mathrm{SNR} &= 10\log_{10}\frac{P_{\mathrm{signal}}}{P_{\mathrm{noise}}} \\ &= 10\log_{10}\frac{\sum_{i=1}^{n} x^2(i)}{\sum_{i=1}^{n}(x'(i)-x(i))^2} \end{aligned} \tag{21}$$

We need to process each dimension separately when implementing CPW on PM resource utilization since each has its feature. As a result, we adopt the calculate method of RMSE, as shown in (22). RMSE is used to measure the degree of difference between the calculated sequence and the original sequence. The lower RMSE, the better performance. If RMSE is large, the calculated sequence is quite different from the original sequence, and much information will be lost. While a smaller RMSE shows it closer to the original sequence, and more information in the original sequence will be kept.

$$\mathrm{RMSE}(\mathrm{X}, \mathrm{h}) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(x^{(i)} - y^{(i)}\right)\right)^2} \tag{22}$$

All the experiments were carried out on a server of Intel(R) Core(TM) i7-5930K CPU @ 3.50 GHz, 62G memory, and 4 GTX TiTan X 12G. The experiment environments are open-source machine learning library Scikit-learn and deep learning framework Pytorch with CUDA 10.0.

When using the Alibaba Cluster Trace dataset, we have found that the changes in net_in and net_out are tiny. As we can see in Figs. 10a and 10b, the net_in and net_out have very small fluctuations during the given period. They did not even change in some specific time. This phenomenon will cause infinite value when calculating SNR using (21).

To solve this problem and evaluate our noise reduction algorithm, we directly used the dimension of CPU utilization in the experiment. We randomly select the resource utilization of 10 physical machines for the experiment each time and record the

Table 1
Noise reduction method comparison result of Alibaba Cluster.

Noise reduction method	Machine:334		Machine:2020		Average	
	SNR	RMSE	SNR	RMSE	SNR	RMSE
SSA	23.1011	2.9780	17.0307	5.6484	20.0103	4.1187
Wavelet	22.3431	3.2496	16.7163	5.8567	19.4022	4.3781
CEEMDAN-WT	29.7308	1.3882	26.1666	1.9730	27.4777	1.6746
CEEMDAN-PE-SSA	27.5247	1.7896	19.6264	4.1893	23.9053	2.6916
CEEMDAN-PE-Wavelet	**33.7634**	**0.8726**	**27.7386**	**1.6464**	**30.3495**	**1.2405**

Table 2
Noise reduction method comparison result of ARM cluster.

Noise reduction method	Machine 1		Machine 2		Average	
	SNR	RMSE	SNR	RMSE	SNR	RMSE
SSA	21.3618	1.2355	7.1316	10.5429	9.0027	8.4030
Wavelet	17.5213	1.9225	19.4762	2.5453	18.7856	2.1453
CEEMDAN-WT	28.4691	0.5400	28.3489	0.9164	28.2682	0.7258
CEEMDAN-PE-SSA	25.8029	0.7341	25.6593	1.2490	25.6455	0.9988
CEEMDAN-PE-Wavelet	**30.7767**	**0.4176**	**33.6641**	**0.4970**	**32.8425**	**0.4249**

average value of 10 the experiments, which is recorded in the *Average* column in Tables 1 and 2. For experiment analysis, due to space limitations, we take Machine334 and Machine2020 (which are randomly selected from our experiment machines due to our limit space) in Dataset A as examples. We randomly selected the physical machine ID 334, 2020 for detailed comparison. We compare our noise reduction algorithm with WT, SSA, CEEMDAN-WT, and a variant of our method called CEEMDAN-PE-SSA. CEEMDAN-WT uses CEEMDAN to decompose the original signal. Then, it uses Wavelet Transform to denoise each IMF from CEEMDAN. CEEMDAN-WT can be considered as ablation of CEEMDAN-PE-Wavelet. In CEEMDAN-PE-SSA, we use SSA to smooth the IMFs instead of Wavelet Transform.

For fairness, we selected the parameters of noise reduction algorithm during the experiment. When using SSA, we select the refactor $RP \in \{4, 5, 6, 7, 8\}$, time window $TW \in \{30, 35, 40, 45\}$. The only one parameter we need to select in Wavelet Transform and CEEMDAN-WT is wavelet threshold $T \in \{0.04, 0.05, 0.06, 0.07, 0.1\}$. The parameters we need to select in CEEMDAN-PE-SSA are embedded dimensions ED, delay time DT, refactor RP, time window TW and the threshold of permutation entropy PT where $ED \in \{6, 7, 8, 9\}$, $DT \in \{6, 7, 8, 9\}$, $RP \in \{6, 7, 8, 9\}$, $TW \in \{30, 35, 40, 45\}$, $PT \in \{0.4, 0.5, 0.6, 0.7\}$.

The parameter we need to select in CPW is ED, DT, PT where $PT \in \{0.7, 0.8, 0.9\}$ and wavelet threshold $T \in \{0.04, 0.05, 0.06\}$. In our experiment, the wavelet base is *db*8. The value range of ED and DT is the same as CEEMDAN-PE-SSA. The comparison result is shown in Table 1.

From Table 1, we can see that our noise reduction method achieves the best result in both SNR and RMSE among the noise reduction algorithm. The average SNR of CPW is 30.3495, which is the highest. In contrast, the average RMSE of CPW is 1.2405, which is lower than other comparison algorithms in this paper. And there is a negative correlation between SNR and RMSE. Typically, the algorithm with bigger SNR will have a smaller RMSE at the same time. CEEMDAN-WT takes second place among the comparison method. Also, we can tell from Fig. 11 that our noise reduction algorithm can preserve the information in the original data as much as possible while smoothing the original data.

Table 2 shows the experiment results of Dataset B. Machine 1 is the controller of our cluster while Machine 2 is one of the members. We use the CPU usage data of these five servers for experiments and record the average value of 10 experiments which is recorded in Average column in Table 2.

From the result of Tables 1 and 2, we can know that CPW achieves the best performance while CEEMDAN-WT ranked second. The performance of SSA in Dataset B is much worse than that in Dataset A. Fig. 12 shows the comparison result among the five noise reduction methods of Machine 1 in dataset B. We can see from Fig. 12 that CPW saves the data at the tip better than the others. It shows that CPW smooths and reduces noise while retaining the original information more than other comparable models.

5.5. Model evaluation results

When evaluating the performance of the model, we compare it with the state-of-art models in time series forecasting. The Seq2Seq framework has recently been more and more effective in natural language processing and time series prediction. In the comparative experiment, we used Seq2Seq-LSTM [39]. This model uses an attention mechanism between the encoder and the decoder. We also used GRU [40] and Seq2Seq-GRU for comparison. We used GRU to replace the LSTM in Seq2Seq-LSTM for Seq2Seq-GRU network to test the Seq2Seq framework's performance in this scenario.

In addition to comparing the framework of sequence processing, we also need to compare the models of multivariate output. DM-LSTM [31] is Deep Multi-output LSTM neural network. In order to illustrate the effectiveness of our denoising algorithm, we combine Wavelet Transform with DM-LSTM as WT-DM-LSTM.

In the previous part, we selected the parameters of the denoise methods. In this experiment, we selected the best-performing parameter combination to experiment. The parameter combination we selected in CPW is $ED = 6, DT = 8, PT = 0.9, T = 0.05$. We first used the networks on the data of one machine for the grid search. After the searching was completed, the best-performing parameter combination was used as the setting parameter for our experiment. From the grid search result, we set the self-feedback coefficient α as 0.65, the number of hidden layers is 3, and the number of neurons in hidden layers is 16. In GRU, TG-LSTM, and DM-LSTM, the number of hidden layers is 3. The number of neurons in hidden layers is 16. In WT-DM-LSTM, the wavelet threshold is 0.05, and the wavelet base is *db*8. For Seq2Seq-LSTM and Seq2Seq-GRU, the experiment setting is the same as above.

In our experiments, the epoch is 500, the learning rate η is 0.05, batch_size is 128, and the dropout rate is 0.5. The regularization coefficient C in our experiments is 1e-5. Since each physical machine recorded a lot of data, we extracted the data from the first 8000 time points for experiments. We randomly selected ten physical machines each time and recorded the average results of CMES after ten experiments in Table 11. The results of Machine 334 and Machine 2020 are shown in Tables 3 and 4. The resource

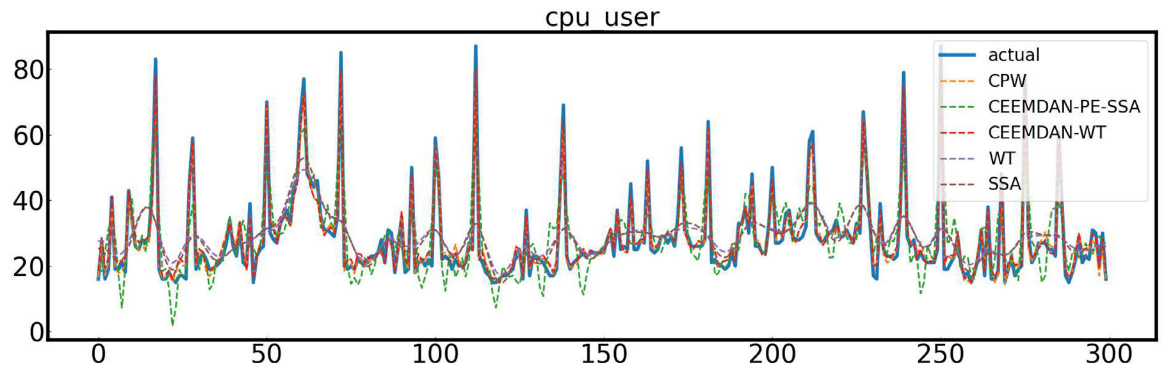


Fig. 11. Denoise result comparison of machine 2020 in Dataset A. Each color represent one noise reduction method.

Table 3
Machine 334 comparison result of dataset A.

Machine_id:334																
Network	Step:3				Step:6				Step:9				Step:12			
	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES
Seq2Seq-LSTM	0.3665	4.4888	0.0705	1.3707	0.0644	4.9697	0.0891	1.8323	0.6214	4.8121	0.0966	1.9033	0.6692	7.1888	0.1187	2.6612
TG-LSTM	2.6612	3.8585	0.0654	1.1628	0.4937	3.7940	0.0755	1.4142	0.4913	4.9156	0.0909	1.7093	0.5323	8.7686	0.1272	13.7267
DM-LSTM	0.2426	3.6757	0.0548	0.9560	0.4170	4.8930	0.0905	1.5664	0.3159	5.2529	0.0777	1.4315	0.4309	5.3986	0.0865	1.7109
WT-DM-LSTM	0.5141	7.7906	0.1126	2.4643	0.3057	3.3748	0.0610	1.0103	0.3733	3.4795	0.0639	1.1455	0.3564	4.5833	0.0682	1.3714
GRU	0.2899	2.8745	0.0532	0.8718	0.6340	8.8743	0.1407	3.0300	0.3691	5.1454	0.0785	1.5240	0.3691	5.1454	0.0785	1.5240
Seq2Seq-GRU	0.3490	4.6142	0.0738	1.3656	0.4320	5.8444	0.0958	1.8199	0.6137	5.3636	0.1044	2.0489	0.6262	5.8674	0.1054	2.2117
ENN	0.1892	2.5882	0.0414	0.6478	0.2262	3.2813	0.0557	0.8496	0.4073	4.0204	0.0744	1.3350	0.3291	4.4288	0.0736	1.2862
CPW-EAMC	**0.1698**	**2.5048**	**0.0382**	**0.5981**	**0.2446**	**2.8058**	**0.0512**	**0.7851**	**0.3205**	**3.6590**	**0.0605**	**1.0984**	**0.3256**	**3.6408**	**0.0621**	**1.1036**

Table 4
Machine 2020 comparison result of dataset A.

Machine_id:2020																
Network	Step:3				Step:6				Step:9				Step:12			
	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES
Seq2Seq-LSTM	0.4674	2.9268	0.0849	1.1371	0.5237	3.6320	0.0960	1.4156	0.4873	3.7720	0.1040	1.4042	0.5167	6.2930	0.1425	2.1256
TG-LSTM	0.4286	3.3383	0.0879	1.1982	0.4996	2.8574	0.0855	1.1568	0.5030	3.0452	0.0903	1.2172	0.5962	4.5812	0.1268	1.8029
DM-LSTM	0.4830	3.0068	0.0886	1.1805	0.5274	2.9321	0.0898	1.2135	0.5169	3.0644	0.0891	1.2402	0.5278	3.8217	0.0977	1.4758
WT-DM-LSTM	0.5300	2.6102	0.0868	1.1169	0.4926	3.0561	0.0909	1.2073	0.5406	3.2300	0.0942	1.3203	0.4063	2.9547	0.0845	1.0650
GRU	0.1590	2.1023	0.0449	0.5112	0.4464	4.2782	0.1026	1.4727	0.5754	4.6986	0.1169	1.8007	0.3589	3.1837	0.0815	1.0557
Seq2Seq-GRU	0.4765	3.3034	0.0938	1.2570	0.4932	3.1321	0.0897	1.2299	0.4786	3.5113	0.0939	1.3176	0.4617	3.3108	0.0888	1.2377
ENN	0.2376	2.2015	0.0538	0.6475	0.1989	3.0471	0.0700	0.7637	0.3241	3.5274	0.0870	1.0842	0.4624	3.5510	0.0905	1.3044
CPW-EAMC	**0.1318**	**2.2080**	**0.0439**	**0.4847**	**0.2325**	**2.4518**	**0.0579**	**0.6949**	**0.2794**	**2.6526**	**0.0651**	**0.8094**	**0.3068**	**3.2497**	**0.0835**	**0.9921**

Table 5
Machine 1 comparison result of dataset B.

Network	Step:3				Step:6				Step:9				Step:12			
	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES
Seq2Seq-LSTM	0.6634	2.4558	0.1907	1.3539	0.9084	3.1559	0.2456	1.7782	0.9441	3.1852	0.2411	1.8052	1.0170	3.2541	0.2466	1.8692
TG-LSTM	0.6918	2.4906	0.1937	1.3831	0.9828	3.2624	0.2411	1.8551	0.8518	3.2364	0.2298	1.7768	0.9270	3.1450	0.2468	1.7829
DM-LSTM	0.7363	2.5619	0.1796	1.4301	0.9422	3.1641	0.2444	1.7969	0.9824	3.2512	0.2417	1.8505	0.9219	3.1512	0.2434	1.7817
WT-DM-LSTM	0.6661	2.3855	0.1906	1.3267	0.9094	3.1328	0.2503	1.7711	0.9108	3.1501	0.2575	1.7821	0.9288	3.2445	0.2397	1.8219
GRU	0.8154	2.6306	0.2170	1.5064	0.9027	2.9327	0.2196	1.6733	1.1015	3.3276	0.2478	1.9376	0.9213	3.1419	0.2432	1.7775
Seq2Seq-GRU	0.7487	2.4344	0.1817	1.4270	0.7985	3.0564	0.2140	1.6717	1.0691	3.7245	0.2456	2.0883	1.2263	4.1183	0.3800	2.400
ENN	0.6653	2.4344	0.1817	1.3437	0.8430	2.9689	0.2220	1.6612	0.9314	3.1396	0.2599	1.7883	11.1818	31.0233	2.4823	24.2579
CPW-EAMC	**0.6734**	**2.3758**	**0.1813**	**1.3237**	**0.8584**	**2.9369**	**0.2287**	**1.6578**	**0.8905**	**3.1384**	**0.2412**	**1.7607**	**0.9066**	**3.0949**	**0.2429**	**1.7511**

utilization forecasting result comparison between the forecasted value of CPW-EAMC and the actual value in 3 steps ahead prediction of Machine2020 is shown in Fig. 13. The horizontal axis is the time axis. We can observe the output of every dimension at each moment since our model is multi-output.

In Tables 3 and 4, RMSE is much larger than MAE and MAPE. Therefore, as mentioned above, we use the square root of RMSE to calculate CMES for maintaining the balance of these three indexes. From the tables, except for the CPW-EAMC, the overall performance of ENN in the experiment is better, while its performance is not stable enough. In Machine 334 and 2020, WT-DM-LSTM performed the best in the LSTM based network in most cases. At step 6 of Table 3, GRU achieved the worst performance, which showed that gradient explosion might happen in this experiment.

Tables 5 and 6 are experiment results of two server in dataset B. We use the same strategy to get the average value of the models' performance as in the previous part. The average results of dataset A are shown in Table 11 while Table 12 records the average results of dataset B. In Table 11, the average CMES of

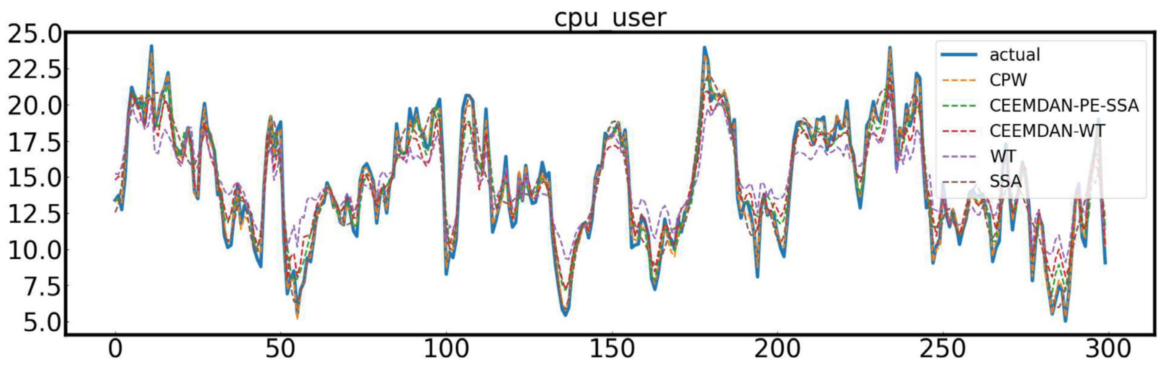


Fig. 12. Denoise result comparison of Machine 1 in Dataset B. Each color represent one noise reduction method. The data shown in this picture is cpu_user.

Table 6
Machine 2 comparison result of dataset B.

Network	Step:3				Step:6				Step:9				Step:12			
	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES
Seq2Seq-LSTM	1.5757	3.2349	0.2795	2.0779	1.9216	3.7200	0.3207	2.4311	1.6231	3.3135	0.2894	2.1339	1.7932	3.5895	0.3073	2.3247
TG-LSTM	1.2013	2.5614	0.2033	1.6177	1.3986	3.0414	0.2465	1.9191	1.6679	3.3514	0.2954	2.1679	1.9045	3.8065	0.3079	2.4622
DM-LSTM	1.2499	2.5821	0.2134	1.6452	1.3995	2.9864	0.2420	1.8920	1.6219	3.3369	0.2726	2.1380	2.1025	4.0834	0.3501	2.6739
WT-DM-LSTM	1.1874	2.5544	0.2042	1.6101	1.4363	2.9889	0.2433	1.9052	1.7566	3.5115	0.2988	2.2726	1.8504	3.6757	0.3133	2.3857
GRU	1.1121	2.3992	0.1899	1.5095	1.4659	3.0237	0.2412	1.9302	1.6380	3.5115	0.2988	2.2726	2.0170	4.1658	0.2946	2.6628
Seq2Seq-GRU	1.3115	2.6435	0.2201	1.6951	1.7230	3.3661	0.2896	2.1887	2.1028	4.2748	0.3360	2.7591	1.9145	3.9130	0.3229	2.5231
ENN	1.1097	2.4062	0.1982	1.5139	1.4672	3.2279	0.2632	2.0344	1.5772	3.3908	0.2742	2.1504	1.6014	3.4050	0.2642	2.1610
CPW-EAMC	**1.0659**	**2.3306**	**0.1894**	**1.4622**	**1.3603**	**2.9276**	**0.2399**	**1.8512**	**1.5550**	**3.3568**	**0.2675**	**2.1247**	**1.7037**	**3.5900**	**0.2943**	**2.2922**

Table 7
Ablation experiment result of Machine 334 in Dataset A.

Machine_id:334																
Network	Step:3				Step:6				Step:9				Step:12			
	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES
EMC	0.2047	2.9008	0.0455	0.7343	0.2943	3.0779	0.0547	0.9243	0.3559	3.7174	0.0644	1.1736	0.4081	4.1035	0.0756	1.3569
EAMC	0.1943	2.6577	0.0406	0.6690	0.2568	2.7965	0.0489	0.8022	0.3573	3.5425	0.0642	1.1349	0.3321	3.8784	0.0710	1.1706
CPW-EAMC	**0.1698**	**2.5048**	**0.0382**	**0.5981**	**0.2446**	**2.8058**	**0.0512**	**0.7851**	**0.3205**	**3.6590**	**0.0605**	**1.0984**	**0.3256**	**3.6408**	**0.0621**	**1.1036**

Table 8
Ablation experiment result of Machine 2020 in dataset A.

Machine_id:2020																
Network	Step:3				Step:6				Step:9				Step:12			
	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES
EMC	0.2294	2.2526	0.0535	0.6471	0.3361	2.6255	0.0712	0.8835	0.3644	2.7963	0.0801	0.9666	0.4557	3.8529	0.0923	1.3751
EAMC	0.1682	2.2658	0.0472	0.5561	0.2369	2.5199	0.0625	0.7174	0.2911	2.6158	0.0655	0.8180	0.3033	3.3507	0.0790	1.0071
CPW-EAMC	**0.1318**	**2.2080**	**0.0439**	**0.4847**	**0.2325**	**2.4518**	**0.0579**	**0.6949**	**0.2794**	**2.6526**	**0.0651**	**0.8094**	**0.3068**	**3.2497**	**0.0835**	**0.9921**

Table 9
Ablation experiment result of Machine 1 in dataset B.

Network	Step:3				Step:6				Step:9				Step:12			
	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES
EMC	0.6998	2.3980	0.1802	1.3453	0.8444	3.1005	0.2608	1.7316	0.9354	3.1259	0.2378	1.7753	0.9144	3.2511	0.2410	1.8182
EAMC	0.7025	2.4640	0.1763	1.3728	0.8988	2.9611	0.2291	1.6866	0.8946	3.1238	0.2496	1.7601	0.9684	3.2202	0.2304	1.8270
CPW-EAMC	**0.6734**	**2.3758**	**0.1813**	**1.3237**	**0.8584**	**2.9369**	**0.2287**	**1.6578**	**0.8905**	**3.1384**	**0.2412**	**1.7607**	**0.9066**	**3.0949**	**0.2429**	**1.7511**

Table 10
Ablation experiment result of Machine 2 in dataset B.

Network	Step:3				Step:6				Step:9				Step:12			
	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES	MAE	RMSE	MAPE	CMES
EMC	1.0864	2.3534	0.1921	1.4801	1.4295	3.0412	0.2527	1.9309	1.6216	3.4556	0.2642	2.1914	1.8115	3.7999	0.3286	2.4408
EAMC	1.0701	2.3388	0.1894	1.4674	1.3849	2.9880	0.2445	1.8890	1.5902	3.4424	0.2747	2.1790	1.7319	3.6603	0.2923	2.3334
CPW-EAMC	**1.0659**	**2.3306**	**2.3304**	**1.4622**	**1.3603**	**2.9276**	**0.2399**	**1.8512**	**1.5550**	**3.3568**	**0.2675**	**2.1247**	**1.7037**	**3.5900**	**0.2943**	**2.2922**

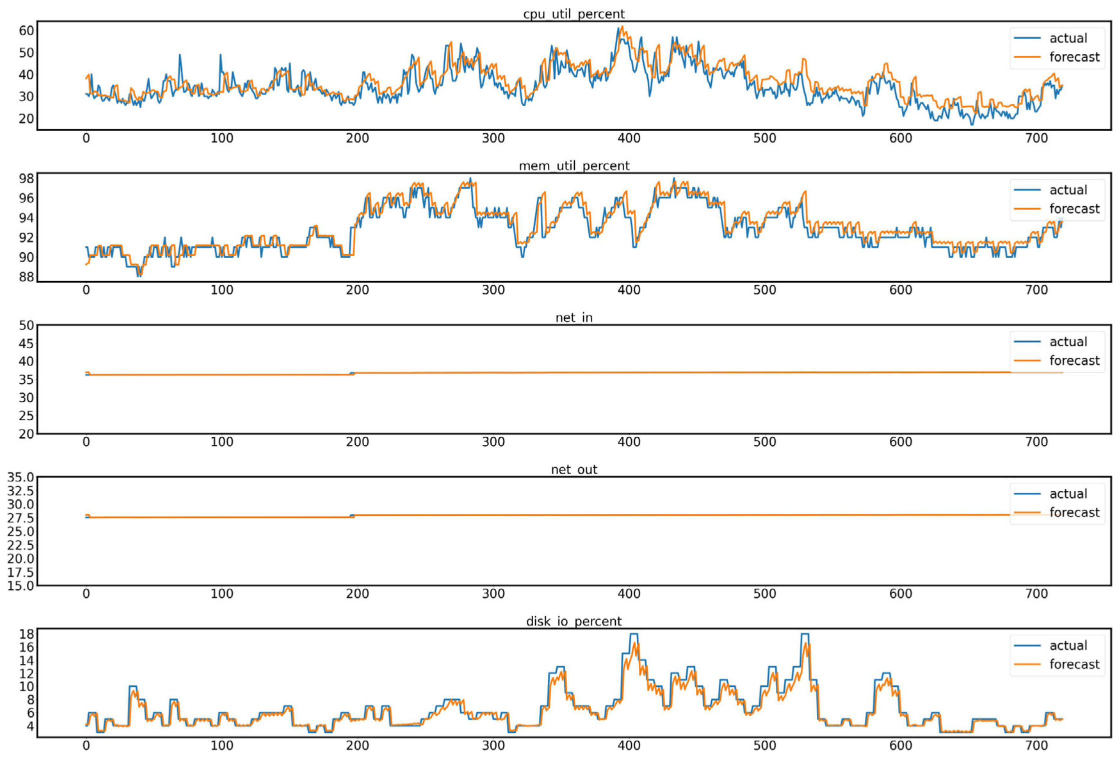


Fig. 13. Three steps ahead forecasting results of machine 2020 with CPW-EAMC at each dimension.

Table 11
Average CMES comparison results of 10 machines in dataset A.

Network	Step:3	Step:6	Step:9	Step:12
Seq2Seq-LSTM	2.3053	2.6857	2.9305	3.2955
TG-LSTM	1.6918	2.0323	2.4364	2.9645
DM-LSTM	1.1779	1.7956	2.7897	2.4955
WT-DM-LSTM	1.8078	2.0876	2.7006	2.6229
GRU	1.2681	2.2486	2.4695	2.6453
Seq2Seq-GRU	2.2645	2.9746	2.8019	2.8996
ENN	10.2754	1.6484	3.8027	2.2610
CPW-EAMC	**1.0053**	**1.3908**	**1.7403**	**1.9354**

Table 12
Average CMES comparison results of 5 machines in dataset B.

Network	Step:3	Step:6	Step:9	Step:12
Seq2Seq-LSTM	1.7181	2.0476	2.0523	2.2003
TG-LSTM	1.6021	1.9903	2.0638	2.1391
DM-LSTM	1.5407	1.9374	2.1048	2.1516
WT-DM-LSTM	1.5845	1.9468	2.0614	2.1341
GRU	1.6705	1.9398	2.0934	2.1702
Seq2Seq-GRU	1.7501	1.9639	2.1893	2.2617
ENN	1.4790	2.3334	2.0672	6.6017
CPW-EAMC	**1.4769**	**1.8436**	**2.0411**	**2.1052**

ENN in step 3 is much higher than other models as the result that ENN performs unstable. Due to the unstable performance of ENN in the ten experiments, the average value is large. CPW-EAMC achieves the best performance in Table 5 at each step. The result of ENN has a substantial error in step 12. In Table 6, however, the result of ENN has fewer advantages over our model in step 12, which makes CPW-EMAC ranked second. This phenomenon shows that the performance of ENN in this scenario is volatile. Although ENN has fewer advantages over our model in step 12 at Machine 2, CPW-EAMC still performs the best average results. The experiment results show that as the number of prediction steps increases, the model's prediction error expands accordingly. It is related to our forecasting strategy. Since we use the circular prediction method in our prediction, the previous prediction error will be passed on to the next prediction. Therefore, the shorter prediction steps, the lower error is reflected by the model.

5.6. Ablation experiment

To show the effectiveness of our model improvement, we conduct ablation experiments on CPW-EAMC. We named the improved model gradually removed as follows: (1) EMC: Remove CPW and attention mechanism from CPW-EAMC; (2) EAMC: Remove CPW from CPW-EAMC. We conduct our ablation experiments on the physical machines in both datasets. As mentioned above, we use the same strategy to record the average performance of models. The average results of ablation experiments in Dataset A are shown in Table 13 while Tables 7 and 8 record the detailed result of Machine 334 and Machine 2020 in Dataset A.

Table 13
Average CMES comparison results of 10 machines in ablation experiment of dataset A.

Network	Step:3	Step:6	Step:9	Step:12
EMC	1.2772	2.6281	2.1727	2.2610
EAMC	1.1837	1.6428	2.0440	1.9798
CPW-EAMC	**1.0053**	**1.3908**	**1.7403**	**1.9354**

Table 14
Average CMES comparison results of 5 machines in ablation experiment of dataset B.

Network	Step:3	Step:6	Step:9	Step:12
EMC	1.4990	1.9242	2.0724	2.1505
EAMC	1.4902	1.8570	2.0623	2.1220
CPW-EAMC	**1.4769**	**1.8436**	**2.0411**	**2.1052**

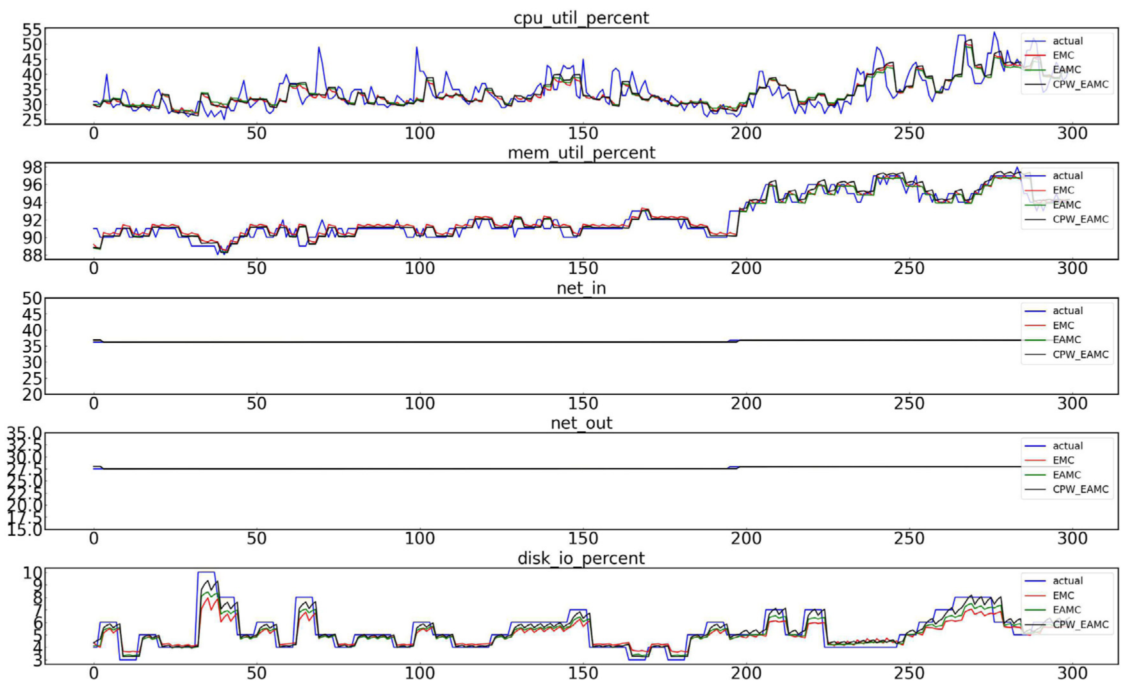


Fig. 14. Ablation experiment comparison result of machine 2020 in three steps ahead prediction. To highlight the details, we only selected 300 data points.

Table 14 shows the average results in Dataset B while Tables 9 and 10 show the detailed ablation experiment results of machine 1 and machine 2 in Dataset B. All tables show a similar trend in ablation experiments: as the prediction steps increase, the performance of models decreases.

We can also conclude from the ablation experiment results that the improved method of our model is adequate and robust. We can see that as our model improves, deviation decrease at each step. The CPW algorithm smoothes the data and improves the robustness and generalization ability of the model. The attention mechanism can fuse the information of the historical state and the current moment to the MLP. While retaining the historical information under long-term dependence, the impact of the current input at the same time is enhanced, thereby improving the model's fitting ability to the data (see Fig. 14).

5.7. Discussion

In this part, we will discuss the cost of models in this paper. With the help of the tool named *Thop*[2] [41], we can analyze the floating-point operations (*flops*), and the number of parameters (*params*) in each model. *Flops* and *params* can be used to calculate the amount of computation of a neural network. To discuss the cost of models concisely, we use an input sample of Dataset B for experiments. Table 15 records *flops*, *params* and *training_time* of the models compared in this paper.

As shown in Table 15, the *flops* and *params* of TG-LSTM is the least while its *training_time* is the longest. The time-consuming operation in TG-LSTM leads to this phenomenon. With the gradual improvement of the model from the original ENN to EAMC, the computational cost increases. Nevertheless, our model is still competitive in terms of overhead compared to other models.

In this scenario, we conducted experiments on ENN and other prediction networks. We found that the prediction accuracy of ENN has an acceptable performance with a short training time. However, the performance of ENN is not stable enough, which is caused by the single-layer neurons of the context layer. With the use of MLP, this network can remember historical features in long-time dependency. Therefore, the network can have a more stable and more accurate prediction. Furthermore, setting a self-feedback coefficient α can effectively focus attention on the recent period, enhancing the impact of recent data and improving the ability to fit sudden changes.

Table 15
Cost of models with input of Dataset B.

	flops	*params*	*training_time* (s)
Seq2SeqGRU	187 416 576	52 012	3308.3
GRU	860 209 152	16 290	4724.2
Seq2Seq-LSTM	302 776 320	66 988	3783.8
TG-LSTM	360 448	258	6935.4
DM-LSTM	57 212 928	21 666	1391.4
ENN	2 998 272	1314	1513.7
EMC	10 862 592	4482	2296.3
EAMC	**21 512 192**	**8738**	**2827.4**

6. Conclusion and future work

We can extract the trend from historical data and predict the future value of PM resources utilization through time series forecasting technology. The resource utilization forecasting of physical machines can provide the scheduling algorithm with future information for scheduling decisions. Therefore, the scheduling algorithm can make a more efficient scheduling decision base on multi-dimensional predictive information.

For the first time, we propose a noise reduction algorithm for processing the utilization of PM resources in a cloud data center and present a MIMO model for PM resource usage prediction. We use CEEMDAN-PE-Wavelet to reduce the noise of the original physical machine resource utilization data. In order to enhance the long-term dependency of ENN, we replace the context layer with a network unit MLP with feature extraction and memory capabilities. To highlight the influence of the input data at the current moment, we adopt an attention mechanism

[2] Thop:Pytorch-Opcounter. https://pypi.org/project/thop/.

so that the network can pay more attention to learning the features of the current moment while maintaining long-term dependence on resource utilization data. Using the physical machine data collected by Alibaba Cluster Trace to evaluate the performance of the CPW-EAMC model, we can see that the model has a performance improvement compared with the current latest time-series processing framework in this scenario.

The resource utilization of physical machines in cloud data centers has a great relationship with the types of submitted tasks by users and the scheduling algorithm of data centers. As time goes by, the historical trends will vary from period to period. It will cause the original forecasting model to become unusable. Therefore, we will focus on adapting our network into an online training model in our future work. With a more accurate prediction model, the scheduling algorithm can make a more efficient scheduling strategy to alleviate the waste of data center resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported in part by the Guangdong Major Project of Basic and Applied Basic Research, China under Project 2019B030302002, in part by the Science and Technology Major Project of Guangzhou, China under number 202007030006, in part by the Industrial Development Fund Project of Guangzhou, China under Project X2JSD8183470, in part by the Engineering and Technology Research Center of Guangdong Province for Logistics Supply Chain and Internet of Things, China under Project GDDST[2016]176, and in part by National Natural Science Foundation of China (62072187, 61872084, 61772205).

References

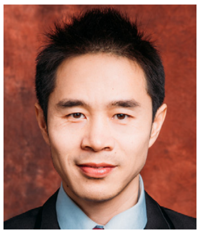
[1] Y. Liu, X. Wei, J. Xiao, Z. Liu, Y. Xu, Y. Tian, Energy consumption and emission mitigation prediction based on data center traffic and pue for global data centers, Glob. Energy Interconnect. 3 (3) (2020) 272–282.
[2] Data centres and data transmission networks, 2020, https://www.iea.org/reports/data-centres-and-data-transmission-networks (Accessed Dec 16, 2020).
[3] Z. Zhou, J. Abawajy, M. Chowdhury, Z. Hu, K. Li, H. Cheng, A.A. Alelaiwi, F. Li, Minimizing sla violation and power consumption in cloud data centers using adaptive energy-aware algorithms, Future Gener. Comput. Syst. 86 (2018) 836–850.
[4] L. Li, J. Dong, D. Zuo, J. Wu, Sla-aware and energy-efficient vm consolidation in cloud data centers using robust linear regression prediction model, IEEE Access 7 (2019) 9490–9500.
[5] F. Liu, Z. Ma, B. Wang, W. Lin, A virtual machine consolidation algorithm based on ant colony system and extreme learning machine for cloud data center, IEEE Access 8 (2019) 53–67.
[6] J. Kumar, A.K. Singh, Workload prediction in cloud using artificial neural network and adaptive differential evolution, Future Gener. Comput. Syst. 81 (2018) 41–52.
[7] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, H. Tenhunen, Utilization prediction aware vm consolidation approach for green cloud computing, in: 2015 IEEE 8th International Conference on Cloud Computing, IEEE, 2015, pp. 381–388.
[8] E. Zharikov, S. Telenyk, P. Bidyuk, Adaptive workload forecasting in cloud data centers, J. Grid Comput. 18 (1) (2020) 149–168.
[9] N.E. Huang, Z. Shen, S.R. Long, M.C. Wu, H.H. Shih, Q. Zheng, N.-C. Yen, C.C. Tung, H.H. Liu, The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis, Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. 454 (1971) (1998) 903–995.
[10] Z. Wu, N.E. Huang, Ensemble empirical mode decomposition: a noise-assisted data analysis method, Adv. Adapt. Data Anal. 1 (01) (2009) 1–41.
[11] X. Cheng, J. Mao, J. Li, H. Zhao, C. Zhou, X. Gong, Z. Rao, An eemd-svd-lwt algorithm for denoising a lidar signal, Measurement (2020) 108405.
[12] M.E. Torres, M.A. Colominas, G. Schlotthauer, P. Flandrin, A complete ensemble empirical mode decomposition with adaptive noise, in: 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2011, pp. 4144–4147.
[13] J. Cao, Z. Li, J. Li, Financial time series forecasting model based on ceemdan and lstm, Physica A 519 (2019) 127–139.
[14] P. Bento, J. Pombo, M. Calado, S. Mariano, Optimization of neural network with wavelet transform and improved data selection using bat algorithm for short-term load forecasting, Neurocomputing 358 (2019) 53–71.
[15] W. Qiao, Z. Yang, Forecast the electricity price of us using a wavelet transform-based hybrid model, Energy 193 (2020) 116704.
[16] Y. Li, Y. Li, X. Chen, J. Yu, H. Yang, L. Wang, A new underwater acoustic signal denoising technique based on ceemdan, mutual information, permutation entropy, and wavelet threshold denoising, Entropy 20 (8) (2018) 563.
[17] A.R.S. Parmezan, V.M. Souza, G.E. Batista, Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model, Inform. Sci. 484 (2019) 302–337.
[18] S. Galeshchuk, Neural networks performance in exchange rate prediction, Neurocomputing 172 (2016) 446–452.
[19] J.T. Connor, R.D. Martin, L.E. Atlas, Recurrent neural networks and robust time series prediction, IEEE Trans. Neural Netw. 5 (2) (1994) 240–254.
[20] Y. Hu, X. Sun, X. Nie, Y. Li, L. Liu, An enhanced lstm for trend following of time series, IEEE Access 7 (2019) 34020–34030.
[21] B. Yang, S. Sun, J. Li, X. Lin, Y. Tian, Traffic flow prediction using lstm with feature enhancement, Neurocomputing 332 (2019) 320–327.
[22] S.O. Sahin, S.S. Kozat, Nonuniformly sampled data processing using lstm networks, IEEE Trans. Neural Netw. Learn. Syst. 30 (5) (2018) 1452–1461.
[23] J. Hu, W. Zheng, Multistage attention network for multivariate time series prediction, Neurocomputing 383 (2020) 122–137.
[24] Z. Lin, L. Cheng, G. Huang, Electricity consumption prediction based on lstm with attention mechanism, IEEJ Trans. Electr. Electron. Eng. 15 (4) (2020) 556–562.
[25] Z. Niu, Z. Yu, W. Tang, Q. Wu, M. Reformat, Wind power forecasting using attention-based gated recurrent unit network, Energy 196 (2020) 117081.
[26] J. Wang, A deep learning approach for atrial fibrillation signals classification based on convolutional and modified elman neural network, Future Gener. Comput. Syst. 102 (2020) 670–679.
[27] Y. Wang, L. Wang, F. Yang, W. Di, Q. Chang, Advantages of direct input-to-output connections in neural networks: the elman network for stock index forecasting, Inform. Sci. (2020).
[28] Y. Zhang, X. Wang, H. Tang, An improved elman neural network with piecewise weighted gradient for time series prediction, Neurocomputing 359 (2019) 199–208.
[29] W. Wu, W. Lin, L. He, G. Wu, C.-H. Hsu, A power consumption model for cloud servers based on elman neural network, IEEE Trans. Cloud Comput. (2019).
[30] Y. Bao, T. Xiong, Z. Hu, Multi-step-ahead time series prediction using multiple-output support vector regression, Neurocomputing 129 (2014) 482–493.
[31] Y. Zhou, F.-J. Chang, L.-C. Chang, I.-F. Kao, Y.-S. Wang, Explore a deep learning multi-output neural network for regional multi-step-ahead air quality forecasts, J. Cleaner Prod. 209 (2019) 134–145.
[32] S. Jeddi, S. Sharifian, A hybrid wavelet decomposer and gmdh-elm ensemble model for network function virtualization workload forecasting in cloud computing, Appl. Soft Comput. 88 (2020) 105940.
[33] M. Kim, J. Jun, N. Kim, Y. Song, C.S. Pyo, Sequence-to-sequence model for building energy consumption prediction, in: 2018 International Conference on Information and Communication Technology Convergence (ICTC), IEEE, 2018, pp. 1243–1245.
[34] H.M. Nguyen, G. Kalra, D. Kim, Host load prediction in cloud computing using long short-term memory encoder–decoder, J. Supercomput. 75 (11) (2019) 7592–7605.
[35] C. Bandt, B. Pompe, Permutation entropy: a natural complexity measure for time series, Phys. Rev. Lett. 88 (17) (2002) 174102.
[36] T. Koskela, M. Lehtokangas, J. Saarinen, K. Kaski, Time series prediction with multilayer perceptron, fir and elman neural networks, in: Proceedings of the World Congress on Neural Networks, Citeseer, 1996, pp. 491–496.
[37] M. Shiblee, P.K. Kalra, B. Chandra, Time series prediction with multilayer perceptron (MLP): a new generalized error based approach, in: International Conference on Neural Information Processing, Springer, 2008, pp. 37–44.
[38] J. Guo, Z. Chang, S. Wang, H. Ding, Y. Feng, L. Mao, Y. Bao, Who limits the resource efficiency of my datacenter: An analysis of alibaba datacenter traces, in: 2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS), IEEE, 2019, pp. 1–10.

[39] F. Li, Z. Gui, Z. Zhang, D. Peng, S. Tian, K. Yuan, Y. Sun, H. Wu, J. Gong, Y. Lei, A hierarchical temporal attention-based lstm encoder-decoder model for individual mobility prediction, Neurocomputing (2020).
[40] F. Shahid, A. Zameer, M. Muneeb, Predictions for covid-19 with deep learning models of lstm, gru and bi-lstm, Chaos Solitons Fractals 140 (2020) 110212.
[41] Y. Wu, Z. Wang, Y. Shi, J. Hu, Enabling on-device cnn training by self-supervised instance filtering and error map pruning, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 39 (11) (2020) 3445–3457.

Yongde Zhang

Fagui Liu

Bin Wang


Weiwei Lin

Guoxiang Zhong

Minxian Xu

Keqin Li