2021 IEEE International Conference on Cluster Computing (CLUSTER) | 978-1-7281-9666-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/Cluster48925.2021.00038

RPTCN: Resource Prediction for High-dynamic Workloads in Clouds based on Deep Learning

Wenyan Chen[1], Chengzhi Lu[1,2], Kejiang Ye[1,*], Yang Wang[1], Cheng-Zhong Xu[3]

[1]*Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences*
[2]*University of Chinese Academy of Sciences*
[3]*State Key Lab of IoTSC, Faculty of Science and Technology, University of Macau*
{*wy.chen, cz.lu, kj.ye, yang.wang1*}*@siat.ac.cn, czxu@um.edu.mo*

***Abstract*—Resource management is challenging in clouds due to the dynamics and sharing characteristics. The crucial problem is how to allocate resources accurately and satisfy demands of workloads timely. The traditional solution is to use historical data to predict future resource usage. Although these resource prediction methods can predict the periodicity, they can not accurately predict mutation points due to the high dynamics and uncertainty of resource usage.**

To tackle this issue, in this paper we propose a resource usage prediction method - RPTCN, which is based on a deep learning method - temporal convolutional networks (TCNs) in cloud systems. We add a fully connected layer and attention mechanism to TCNs to improve the prediction accuracy. In order to explore the relationship between the usage of different resources in the temporal dimension, we use correlation analysis to screen performance indicators as multidimensional feature input for prediction. Finally, we evaluate the performance of this method on Alibaba trace v2018. Evaluations show that RPTCN improves the overall MAE and MSE by 6.50%∼89.03% and 0.41%∼68.82% respectively compared to baselines in dynamic and long-term prediction of resource usage. Moreover, the convergence and generalization of RPTCN are also better than the baselines.

***Index Terms*—cloud computing, resource prediction, high dynamic, deep learning**

I. Introduction

With the rapid development of cloud computing technology, cloud services have gradually penetrated into the daily life of end-users [1], [2]. Cloud computing platforms can provide users with shared resources in a "pay-as-you-go" manner, and also can timely respond to user requests, which improves the service experience of users. However, the resource demands of workloads change constantly over time and may fluctuate violently. In addition, co-locating different workloads in the same server may lead to performance loss caused by interference. *Resource fragmentation*, *resource overbooked* and *unbalanced resource utilization* [3]–[6] occur frequently, causing resource under-utilization of cloud clusters. How to allocate resources to workloads dynamically and reasonably in order to improve cluster performance while reducing the resource waste is a big challenge confronted by cloud operators.

Dynamic resource allocation relies on accurate prediction of future resource usage, and cloud operators can make reasonable allocation through the estimated values of some resource. Therefore, accurate resource prediction can effectively reduce the waste of resource, while providing users with high service quality. By doing so, it can also greatly reduce the physical equipment expenditures of cloud service providers, and maximize the usage of shared resources.

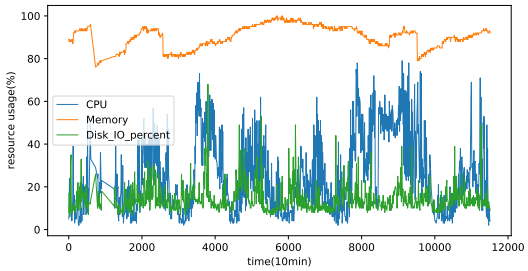


Fig. 1. Different resource utilization of workloads on containers in Alibaba cloud cluster.

We conduct a preliminary characteristic analysis of Alibaba cluster trace [7] and find that the resource utilization of this cluster shows obvious *uncertainty*, as shown in Fig. 1. The CPU usage, memory usage and disk I/O fluctuate significantly and represent no regularity for a long time period in this figure. High-dynamic workloads resource utilization can bring some problems for resource allocation, such as idle resources due to over-allocation of resources and degraded workloads performance due to under-allocation of resources. Therefore, it's important to predict the dynamic resource usage accurately.

Recently, researchers have proposed some resource usage prediction models suitable for time series [8]–[11]. However, most of these models are based on one-to-one mode, which means the data fed into the predictor only includes the historical information specific to the predicted resource, such as CPU usage sequence for CPU usage prediction. We argue that this prediction mode is not always efficient as the CPU usage is not only related to its historical log but also to the historical memory usage, disk read/write rate, cache miss and IPC (instructions per cycle), etc. Therefore, how to predict resource usage through multi-dimensional indicators is the focus of this paper.

As we find that the resource usage of workloads in the real

*Corresponding author.

978-1-7281-9666-4/21/$31.00 ©2021 IEEE
DOI 10.1109/Cluster48925.2021.00038

cloud cluster no longer exhibits classic periodic characteristics but presents obvious sudden changes, we need to design a new model to adapt to this case. Deep learning [12] provides new ideas for time series prediction. Different deep neural networks (DNNs) have been successfully applied in computer vision, speech recognition, natural language processing, and so on. The powerful computing and learning capability of DNNs provides a good basic model for dynamic resource usage prediction. Many researchers usually use recurrent neural networks (RNNs) to solve the time-series prediction problems, and they design model structures suitable for specific scenarios according to different target requirements. Recently the emergence of temporal convolutional networks (TCNs) based on convolutional neural networks (CNNs) provides a new opportunity for time-series prediction. This method has obvious advantages in accuracy and convergence.

In this paper, we propose a new prediction model - RPTCN, to address the dynamic and long-term dependence of cloud resource usage. RPTCN is an optimized model based on TCNs that uses a fully connected layer and attention mechanism to improve resource usage prediction accuracy in cloud clusters. In addition, we expand the input feature dimensions horizontally in time series to strengthen the weight of short-term dependence. The main contributions of this paper are as follows.

- We propose a new method RPTCN for resource prediction in cloud clusters based on TCNs model, and add a fully connected layer and attention mechanism on TCNs to improve the prediction performance.
- We combine dependencies and correlations among multidimensional resources, and expand the input feature dimensions horizontally in time series in order to capture the weight and short-term dependence information of performance indicators that have a strong correlation with the predicted resource.
- We compare and evaluate the prediction performance of RPTCN with the baselines on real workloads in Alibaba cluster. Experimental results show that RPTCN has a higher prediction accuracy than state-of-the-art prediction models in the dynamic and long-term prediction of resource utilization. Furthermore, the convergence and generalization of RPTCN are also outperforming the baselines.

The rest of this paper is organized as follows. We discuss the background and motivation of this research in Section II. We give an overview of our model in Section III and present the experiment design in Section IV. After that, we evaluate the performance of the proposed model in Section V and also provide an analysis of the related work in Section VI. Finally, we conclude this paper in Section VII.

II. Background and Motivation

Cloud computing with large-scale datacenters is a popular industrial computing paradigm, given a cost-efficiency and the pay-as-you-go service mode for end-users. However, as the scale of datacenters grows up, low resource utilization becomes a crucial concentration for cluster administrators to achieve efficient system performance.

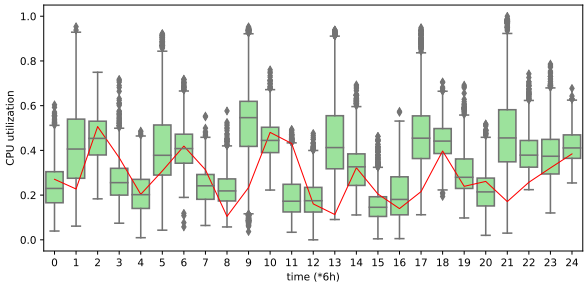


Fig. 2. The boxpot of CPU utilization distribution each 6 hours (The red line means the average CPU utilization in 6 hours interval).

Co-locating different types of workloads (such as online services and offline workloads) on shared resources in large-scale clusters to improve resource utilization thus becomes a common practice in product cloud clusters [13]–[18]. However, this deployment strategy still has great *imbalance* [5] in resource allocation and job scheduling, which is mainly reflected in two aspects.

1) **The server resources in the cloud cluster are heterogeneous, diverse and shared.** In typical cloud clusters, there are many types of resources and large differences among their functions. Not only are the software and hardware configurations various, but the access interfaces and management rules are different. The resource usage of servers remains 40%∼60%, leading to many idle resources. In addition, the uneven resource usage of cloud clusters is mainly reflected in these aspects as follows. The different types of hardware resources are unevenly used, the distribution of workloads on different nodes is *uneven* and *imbalanced* in different time dimensions. How to make full use of the advantage of heterogeneous hardware resources to improve resource utilization has become a prominent challenge.

2) **The resource demands of workloads in clouds are dynamic and complex.** Workloads in cloud clusters usually are co-located with each other, and the patterns of users accessing these cloud services, including time-frequency and spatial distribution, are very different. Applying for these resources for different tasks at the same time will inevitably lead to resource conflict and interference, which results in unpredictable performance loss as evidenced by our previous work [19]. At the same time, the complex associations existing among workloads will evolve over time, resulting in uncertainty of resource requests. If these unpredictable requests are not handled efficiently, it will not only reduce the system performance, affecting user experience but also increase the management overhead of cloud clusters. How to deal with the different requirements of diverse workloads and ensure their respective quality of service and minimize the impact of interference has become another challenge.

CPU resource is one of the most precious computing

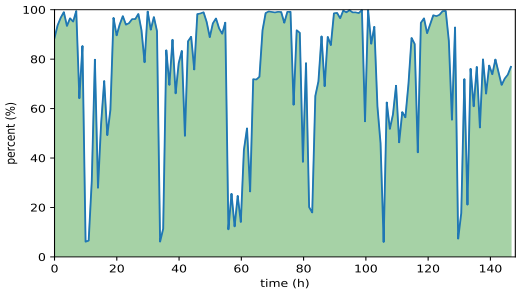


Fig. 3. The percent distribution of machines that used less than 50% CPU resource of all machines.

resources in cloud clusters. In order to be able to use CPUs efficiently, clusters managers usually co-locate different workloads on the same server to share CPU. However, the analysis of Alibaba trace v2018 shows that a large amount of CPU resource remained idle even though the workloads are co-located [20].

Fig. 2 depicts the average CPU usage of the workloads over time in Alibaba cloud cluster. The red line in the figure takes a sampling point every six hours, which represents the change of the average CPU usage throughout the sampling time. The boxplot shows the distribution of the average CPU usage of the cluster between two sampling points. As can be seen from the red line, the average CPU usage has a certain periodicity. The upper quartile of the boxplot at each sampling point is mostly less than 0.6. Therefore, it can be known that there are 75% of the time that the average CPU usage of the cluster is less than 0.6. This indicates that the low CPU resource usage is the common state of the cluster, and the cluster resource usage remains low for a long time.

We further analyze the reasons for the low average resource usage in this cluster. As shown in Fig. 3, by counting the percentage of machines with less than 50% CPU usage, we find that the majority of machines in the cluster are less than 50% CPU usage in most time periods. In addition, more than 80% of the machines maintain CPU usage below 50%. This further proves that the inefficiency in resource allocation and job scheduling in the cluster.

We suspect that one of the important reasons for the above phenomenon is that the resource manager cannot accurately predict the possible CPU usage in the future. In order to ensure the normal execution of the workloads, the resource manager should reserve reasonable resource to satisfy peak usage.

Thus, workload prediction is an important process in cluster management. The predictive result can provide support for job scheduling and an effective reference for resource allocation, thereby ensuring the efficient execution of workloads in cloud systems and providing end-users high service quality.

However, most of the existing workload prediction models use off the shelf algorithms or integrated models, lacking in-depth analysis of actual cloud cluster scenarios. As is known, the model structure and input data have a strong relationship with the prediction accuracy. Due to the resource usage characteristics of different application scenarios are various, the generalization of existing prediction models need to be improved, and there are fewer prediction models for multi-dimensional data to predict a certain resource. Due to the rapid development of deep learning, we plan to use deep learning to design an efficient prediction model that is suitable for dynamic cloud clusters.

III. Model Overview

In the following, we will describe our proposed method RPTCN and explain each step in detail.

A. Data Preprocessing

Generally, the dataset is partially incomplete or has outliers due to network anomalies, system interruption etc. Therefore, we preprocess the dataset before the model is trained to ensure the integrity of the input data. We first screen the records with complete information from the trace. Then we perform data normalization processing (as eq. (1)) in order to eliminate the dimensional impact among indicators.

$$x_{norm} = \frac{x - X_{min}}{X_{max} - X_{min}} \tag{1}$$

Where x represents a value of the time series sequence $X = \{x_1, x_2, ..., x_n\}$, X_{max}, X_{min} are the maximum value and the minimum value of X.

B. Correlation Analysis

There is a certain correlation between the predicted resource and other performance indicators. We use pearson correlation coefficient (PCC) analysis to rank these correlations (PCC is a linear correlation coefficient that reflects the degree of correlation between two variables.). eq. (2) is the calculation method of PCC, where $\rho_X \rho_Y$ is the product of the standard deviations of X, Y. μ_X is the average of X, μ_Y is the average of Y, and E is the expectation.

$$\rho(X, Y) = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\rho_X \rho_Y} \tag{2}$$

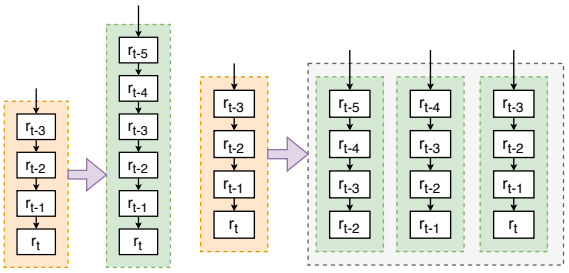
Furthermore, the top half of all performance indicators in the rank list are used as the input data.

C. Data Expansion

In addition to using more performance indicators, we can also extend the existing performance indicators to improve the prediction accuracy of our model.

One common method is to expand the input data vertically, that is, to use longer time-series information for each performance indicator, as Fig. 4(a) shows. The vertical expansion of input data can increase the long-time dependence information input into the model, which is helpful to improve the accuracy of the model when predicting the periodic resource usage of the real-world workloads. However, this method will cost more time on training model to some degree.

Another method is to horizontally expand the existing performance indicators in the time dimension. As shown in



(a) Method A (b) Method B

Fig. 4. Different feature expansion methods (r_t means the usage of some performance indicator r at time t).

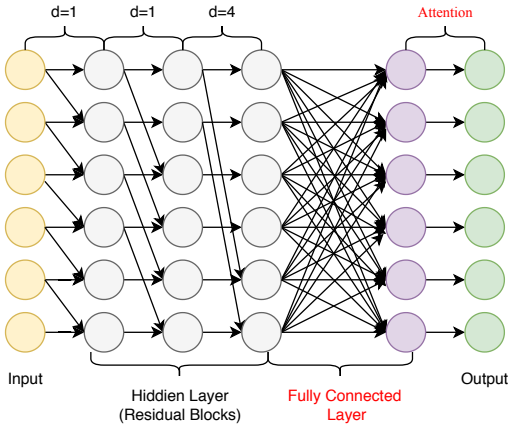


Fig. 5. This is the architecture of RPTCN, and its filter size=3, dilations=[1,2,4].

Fig. 4(b), for the same time series as the previous method, we extend the performance indicator r to three time series sequences, with a sampling interval between each time series. We can see that although the length of the input data is not increased in the vertical direction, through horizontal expansion, we also increase the time interval of the input data (from $[r_{t-3}, r_t]$ to $[r_{t-5}, r_t]$) so that the model can obtain more short temporal information. In addition, it can also increase the weight of the performance indicator at some moments to the prediction result. Expanding the performance indicators horizontally in the time dimension also generates short-term time-series information in the horizontal direction of the dataset, thereby increasing the weight of the prediction result at short-term neighboring moments.

Therefore, we use method B in Fig. 4(b) to expand the input data dimensions in order to feed more time-series information into prediction model.

D. Method Design

1) Model architecture: Time series prediction generally uses RNNs due to the structure of RNN's cyclic autoregression is a good representation of time series. At the same time, with the help of the state memory unit, RNN-based models can associate the output with all input sequences. However, different from voice and text data, there are more complicated influencing factors for CPU usage. Therefore, analyzing time-series relationships of CPU usage only with itself lacks reliability. When considering that CPU usage is affected by other factors such as *IPC*, *MPKI (misses per kilo instructions)* etc, CNNs for multi-dimensional features are undoubtedly the common choice for designing prediction models.

However, traditional CNNs are generally considered to be not suitable for modeling a long-time series problem now. This is mainly due to the limitation of its convolution kernel, which cannot fully capture the long-term dependence information contained in the sequence. To solve this problem, a causal CNN has been proposed. The typical representative model is WaveNet [21]. However, causal CNN also has some limitations. A causal CNN requires a very large number of layers and huge convolution kernels to obtain a wider receptive field which is necessary for the formal construction of long-term memory. The proposal of dilated convolution overcomes the problem of causal convolution. By omitting some hidden layer data to achieve a larger step size, the expansion convolution can use a very small number of layers to obtain a larger receptive field.

A typical model for applying dilated convolution is TCNs model. It combines the modeling ability in the time domain and the feature extraction ability with low parameters of convolution. It was first proposed by Bai et al. [22] to solve the problem of segmentation and detection of human action fragments in videos. TCNs solve the problem of network degradation by using the residual module. It can not only use the information of resource usage in the time domain but also fully consider the impact of other performance indicators on the predicted resource.

On the basis of TCNs, we add a fully connected layer and attention mechanism as shown in Fig. 5. The purpose of adding a fully connected layer is to linearly combine the features extracted by the previous convolution layer to synthesize the impact of different feature values on resource utilization. Afterwards, the attention mechanism is used to adjust the weights of the performance indicators at different moments to the predicted CPU usage, so as to predict future values more accurately.

Fig. 5 shows a RPTCN architecture with a kernel size 3 and dilations $[1, 2, 4]$. RPTCN takes a sequence $(x_0, x_1, ..., x_t)$ as input, and the corresponding $(y_0, y_1, ..., y_t)$ as the expected output. As mentioned above, the important components of RPTCN are causal convolution, 1D fully-convolutional network (1D FCN) [23], dilated convolution, residual block [24], fully connected layer, and attention mechanism. Then we describe these components in detail.

Causal Convolution. Causal convolution restricts the convolution operation to an interval so that it can only operate on inputs from past several steps, and this ensures that future information does not leak into the past. For one-dimensional convolution, causal convolution can be achieved by simply

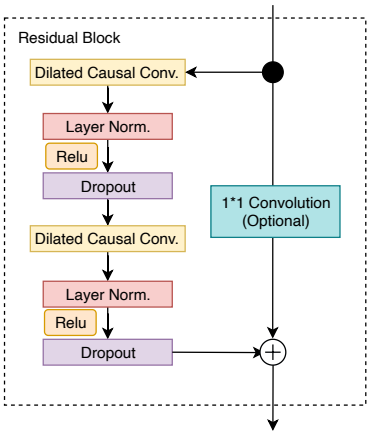


Fig. 6. This is a sample of a residual block in RPTCN architecture.

moving the output of a general convolution by several time steps. The prediction result of current time t is only related to historical data. We define filter as $F = \{f_1, f_2, ..., f_K\}$ (K is the filter size), and a sequence $X = \{x_1, x_2, ..., x_T\}$. Then the causal convolution at x_t is represented as eq. (3)

$$(F * X)_{x_t} = \sum_{k=1}^{K} f_k x_t - K + k \tag{3}$$

1D FCN. In TCNs, different hidden layers are connected using FCN. Since different layers have different kernel size, TCNs need add zero-padding of length (kernel size - 1) to keep subsequent layers the same length as previous layers, so as to make the output correspond to the input one by one in the multidimensional space.

Dilated Convolution. Dilated convolution overcomes the petty receptive field issue of causal convolution. As the quantity of layers increases, the interval between convolution units increases exponentially in order to obtain more long-term dependence information. The dilated convolution with a dilatation rate d at x_t is denoted by eq. (4).

$$(F *_d X)_{x_t} = \sum_{k=1}^{K} f_k x_t - (K - k)d \tag{4}$$

Therefore, the output y_t can be represented as $y_t = f_1 x_{t-2d} + f_2 x_t - d + f_3 x_t (d = 2)$. The receptive field size of the dilated convolution is $(K-1)d+1$, thus increasing K or d can increase the receptive field.

Residual Block. Residual linking is an effective method for training stable TCNs. Instead of convolution layers, TCNs use residual blocks to avoid network degradation with the increased network depth. As shown in Fig. 6, a residual block includes two branches. One branch changes the input x through a number of neural network layers F, including the dilated causal convolution with the weight normalizing layer, the activation function (Relu), and the spatial dropout layer for regularization. Another branch adds a 1×1 convolution of the input data to the residual function to make the length of the output equal to the input. Thus, the output o of a residual block is:

$$o = Activation(x + F(x)) \tag{5}$$

where $Activation$ denotes the activation function.

Fully Connected Layer. The fully connected layer can perform a weighted summation of the learned feature values to achieve the extraction of key features. Its essence is a linear transformation from one feature space to another. The core of the fully connected layer is as eq. (6), where x represents input data, W represents weight, b represents offset, and y represents output.

$$y = \mathbf{W}x + \mathbf{b} \tag{6}$$

Attention Mechanism. The goal of attention mechanism is to select the most critical information for current task goals from the numerous features. There are many different structures of attention mechanisms, such as Bahdanau Attention [25] and Luong Attention [26]. In general, the implementation of attention is:

$$\mathbf{a} = \mathbf{f}_\phi(\mathbf{x}) \tag{7}$$

$$\mathbf{g} = \mathbf{a} \odot \mathbf{z} \tag{8}$$

where $\mathbf{x}$ is the input data, $\mathbf{z}$ is a feature vector, $\mathbf{a}$ is an attention vector, $\mathbf{g}$ is an attention glimpse, $\odot$ means element-wise multiplication and $\mathbf{f}_\phi(\mathbf{x})$ means an attention network with parameters ϕ.

2) Pseudocode: These above observations inspire the design of an effective prediction model in a resource manager. We describe it as algorithm 1.

In algorithm 1, we first declare the history logs of performance indicators as data input ($R_{src} = \{r_1, r_2, ..., r_m\}$) and the future CPU usage as output ($cpu_{m+1}, cpu_{m+2}, ..., cpu_{m+k}$) of the model, and then describe the algorithm workflow in detail.

As the algorithm shows, firstly, we clean the data set to obtain the complete data, then, the data could be normalized

Algorithm 1 Dynamic Resource Prediction

Input:
The logs of performance indicators from time 1 to m: $R_{src} = \{r_1, r_2, ..., r_m\}$, where $r_i = cpu_i, mem_i, ...$

Output:
CPU usage: $cpu_{m+1}, cpu_{m+2}, ..., cpu_{m+k}$

1: $R_{avail} = \{r_1, r_2, ..., r_d\} \leftarrow DataClean(R_{src})$
2: $R_{norm} \leftarrow Normalized(R_{avail})$
3: $p \leftarrow length(r_i)/2$
4: Get the top p performance indicators related to CPU usage:

$$r_i^{'} = \{cpu_i, ..., perf_p\}, {R_{norm}}^{'} = \{r_1^{'}, r_2^{'}, ..., r_d^{'}\}$$

5: $R_{extend} \leftarrow DataExpansion({R_{norm}}^{'})$
6: $cpu_{m+1}, cpu_{m+2}, ..., cpu_{m+k} \leftarrow RPTCN(R_{extend})$
7: return $cpu_{m+1}, cpu_{m+2}, ..., cpu_{m+k}$

(max-min method), and next we use PCC analysis to obtain the top p ($length(r_i)/2$) indicators with the highest correlation with the CPU usage. The next step is to use data expansion method (in Fig. 4(b)) to expand the data dimensions horizontally. Finally, the preprocessed data is input as features into RPTCN model for training and cross-validation, Finally, we can get the predicted CPU usage sequence.

IV. Experimental Design

In this section, we introduce the experimental environment setup, dataset, baselines and evaluation metrics that we used.

A. Experiment Setup

The experimental environment used in this paper is a server configured with four hardware accelerator Tesla-V100 GPUs. The machine learning library is Tensorflow-gpu v1.14.0 and Keras v2.3.1.

During the model training process, in order to avoid overfitting, we use the callback function EarlyStopping to prevent model overfitting, and the parameter *patience* is 10.

B. Dataset

We evaluated the prediction model using Alibaba trace v2018 [7]. The trace, released in November 2018, records offline workloads and online services from a cluster of 4034 machines overall an 8-day period. Almost all machines in this cluster are co-located online services and offline jobs simultaneously.

Due to the resource sharing nature of Alibaba cloud cluster, the running process of jobs is affected by resource competition and performance interference, and the resource utilization of jobs shows an obvious imbalance and fluctuation. We use RPTCN to conduct experiment on the resource usage of physical machines and containers in this cluster, to verify the accuracy and generalization of RPTCN.

In the following experiments, we set the time interval to 10s, and the dataset is divided into three parts: training, validation and test data, with a ratio of 6:2:2, which is a common ratio in time-series data.

C. Baselines

We evaluate the performance of our prediction model RPTCN by comparing it with several commonly used time-series prediction methods. They are ARIMA [27], XGBoost (It performs well in Kaggle contests), LSTM [28] and CNN-LSTM [29].

D. Evaluation Metrics

To quantify and compare the performance of RPTCN with the four baselines, we use two different performance metrics as follows.

- MSE (Mean Square Error)

$$MSE(y_i, \hat{y}_i) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{9}$$

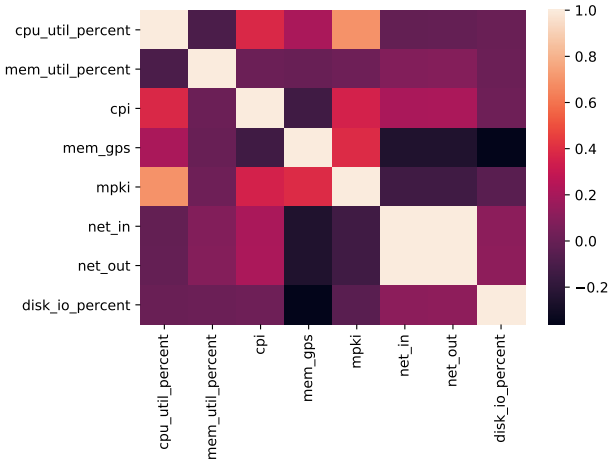


Fig. 7. Correlation analysis of the indicators in container c_18104. The x and y-coordinates are the names of the monitoring indicators for the sampling. The specific meaning of each indicator is described in Table I.

- MAE (Mean Absolute Error)

$$MAE(y_i, \hat{y}_i) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{10}$$

In eq. (9) and eq. (10), y_i is the true value, $\hat{y}_i$ is the predicted value, and n is the length of all true values.

V. Performance Evaluation

A. Data Preparation

In order to filter these indicators, we perform correlation analysis between CPU utilization and other indicators, and the indicators are described in Table I. The result is shown in Fig. 7. The lighter the color, the stronger the correlation. It can be found that the top four indicators which have a stronger correlation with CPU utilization are *cpu, mpki, cpi, mem_gps*.

The next step is to expand the data dimensions horizontally. According to the results of the top four indicators obtained in the previous step and the method of data expansion, the features of the input data are as follows (take the container c_18104 as an example).

$$R_{input} = R(cpu_e, mpki_e, cpi_e, mem\_gps_e) \tag{11}$$

TABLE I
The Meaning of Each Indicator

Indicator	Meaning
cpu_util_percent	cpu utilization percent
mem_util_percent	memory utilization percent
cpi	cycles per instruction
mem_gps	normarlized memory gigabyte per second
mpki	misses per kilo intructions
net_in	normarlized in coming network traffic
net_out	normarlized out going network traffic
disk_io_percent	disk io percent

TABLE II
THE EXPERIMENTAL RESULT OF ACCURACY ON ALIBABA TRACE

Model		Containers		Machines	
		MSE (10^{-2})	MAE (10^{-2})	MSE (10^{-2})	MAE (10^{-2})
Uni	ARIMA	0.4627	5.0581	0.5662	5.9978
	LSTM	2.8443	14.1424	4.5306	15.4407
	CNN-LSTM	0.6876	6.7262	0.6961	6.9056
	XGBoost	0.3833	4.7039	1.2931	9.2153
	RPTCN	0.3519	4.3164	0.7046	6.4442
Mul	LSTM	0.4276	4.9460	0.8271	7.3212
	XGBoost	0.4138	4.8611	9.4901	22.2577
	CNN-LSTM	0.3983	4.8108	2.0102	11.1652
	RPTCN	0.3849	4.6580	0.7597	6.8608
Mul-Exp	LSTM	0.3169	4.1077	2.2257	11.9627
	XGBoost	0.3274	4.2841	4.4529	16.1577
	CNN-LSTM	0.3402	4.3305	2.8865	13.4577
	RPTCN	**0.2963**	**4.0910**	**0.4884**	**5.0386**

In eq. (11), $cpu_e = cpu_{t-2}, cpu_{t-1}, cpu_t$, the same as $mpki_e$, cpi_e, and mem_gps_e.

B. Result and Analysis

We set the input data of our experiment to three scenarios with different data dimensions, which are univariate, only CPU utilization as input (*Uni*), the top half of all indicators (*Mul*), and the top half of all indicators with time dimension horizontal expansion (*Mul-Exp*).

1) Accuracy: The accuracy of model prediction is an important indicator for evaluating our model. It can indicate whether RPTCN has learned valuable information on time-series data. In addition, we also evaluate the generalization of RPTCN on the monitoring logs of the workloads running on containers and physical machines. The results are shown in Table II and Fig. 8.

Firstly, we describe the result of the univariate scenario (Uni). Uni scenario is the most widely used in time-series prediction, because the trend of some resource utilization has the strongest correlation with itself, and can best reflect the change characteristic of this resource utilization over a time period (but in fact, changes in other performance indicators can also provide prior valuable information for this resource prediction). It can be seen from Table II that RPTCN performs best in predicting the resource utilization of containers, while ARIMA performs best in predicting the resource utilization of physical machines. That is probably because the ARIMA model mainly considers the difference between adjacent time intervals when it is training.

In the second multivariate scenario (Mul), we compare our RPTCN with LSTM, XGBoost, and CNN-LSTM. As can be seen from Table II, the MSE and MAE of RPTCN are the smallest. However, these values are relatively bigger when compared with the values of the univariate scenario. Therefore, we speculate that the use of too many dimensions of input may be not helpful in predicting CPU utilization, and variables with weak correlation with CPU utilization will increase the training time of the model and memory space of physical machines. Therefore, in order to optimize the prediction model, some methods must be used to prune the input variables in multi-dimensional time-series prediction. For CPU resource prediction in cloud clusters, the period of historical data collected is usually as long as several weeks or months, so the data size is huge, and the prediction result needs to be used for real-time dynamic resource allocation. Therefore, the quality and size of the input data should be optimal. In addition, we can't ignore the importance of performance indicators which have a strong correlation with CPU resource to be predicted. They can provide a good reference for the prediction of future CPU utilization. In consideration of these aspects, we design the experimental evaluation of the third scenario.

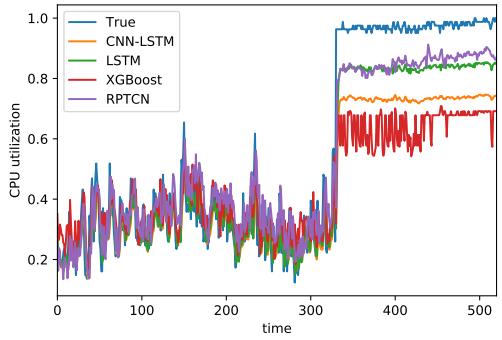


Fig. 8. The predicted value vs true value of RPTCN compared with baselines.

In multivariate expansion (Mul-Exp) scenario, XGBoost, CNN-LSTM, and RPTCN perform similar on containers with MSE and MAE, but when these models are trained on machines, RPTCN shows ordinary performance. At the same time, we can see that compared to the Uni and Mul scenarios, MSEs and MAEs of different models are all better in predicting the resource utilization of containers. But when predicting the resource utilization of machines, LSTM-based models have some performance degradation in Mul-Exp scenario, and RPTCN has the best accuracy on machines. We speculate that when using multidimensional data, more interference is introduced compared to Uni scenario, but by extending the data dimensions horizontally, the model will obtain more information to offset the interference, and even fit the dynamic trend of CPU through the information of multidimensional data, so some models with Mul-Exp scenario have better results compared to Uni, and RPTCN performs the best in this fitting process.

In order to see the prediction effect more intuitively, we draw the curve graph between the real value and the predicted value of the Mul-Exp scenario. As shown in Fig. 8, when the workload is running on the physical machine, there is no specific regularity for the change of CPU resource utilization,

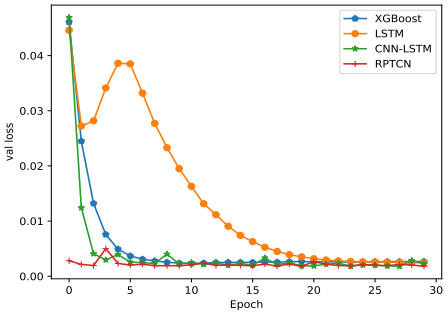


Fig. 9. Loss convergence of RPTCN and baselines when they are trained on the containers of Alibaba trace v2018

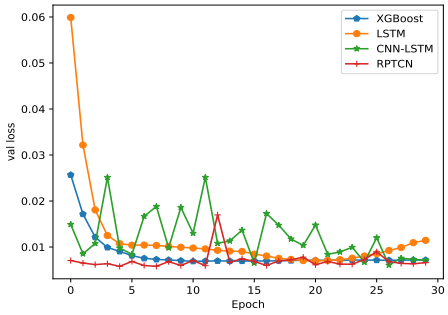


Fig. 10. Valid loss of RPTCN and baselines when they are trained on the machines of Alibaba trace v2018

and the CPU resource utilization increases abruptly after the 350th sampling point, and then maintains a high CPU resource utilization. Although the baselines can predict the sudden increase trend, the predicted value and true value differ greatly, and the predicted values have not been corrected since then. However, RPTCN can accurately predict the range of sudden increase, which indicates that RPTCN has relatively high accuracy in predicting this kind of dynamic time-series data, which is in line with our expectations.

2) Convergence: Besides accuracy, the speed of loss convergence is also an important metric to measure the performance of a deep learning model. In order to evaluate the performance of RPTCN from multiple perspectives, we compare the loss convergence of RPTCN with the baselines in this part.

Fig. 9 and Fig. 10 show the loss convergence results of different models on containers and physical machines. As can be seen from Fig. 9, the loss value of RPTCN is very small at the beginning, while the loss value of other models is relatively large, and LSTM suddenly increases when $epoch = 5$. We speculate that this may be caused by overfitting of LSTM. XGBoost and CNN-LSTM have relatively smooth loss curves, while RPTCN has always maintained a small loss value, which is better than these baselines.

In Fig. 10, it can be clearly seen that the loss curve of CNN-LSTM has an unusual jitter, which makes the convergence rate of CNN-LSTM slow down. It can be seen that the CNN-LSTM model gradually converges after 25 epochs. We speculate that the LSTM layer doesn't match the real batch data distribution during training. RPTCN keeps a very small loss value as that on containers while the loss value of LSTM is the largest at the beginning, but it also converges rapidly after several epochs, and most of them are higher than that of RPTCN. The initial loss value of XGBoost is less than LSTM but larger than that of RPTCN and CNN-LSTM.

Therefore, the loss of RPTCN can converge more quickly to a smaller value than the baselines on containers and machines, which indicates that its performance is better than the baselines.

C. Discussion

Through the evaluation of the prediction accuracy and loss convergence of RPTCN, we can find that TCNs give full play to their advantages and perform well when comparing with baselines. Our model not only has a high accuracy but also has a significant advantage in convergence speed. We mainly focus on the prediction of CPU resource with dynamic changes and unpredictable mutation points. We prove that RPTCN has a small difference between the predicted value and the real value when predicting resource utilization of such a dynamic scenario. Moreover, through the verification of resource utilization of the workloads running on machines and containers, we can see that the model has good generalization and can be widely used in similar resource prediction scenarios. The cluster manager can allocate dynamically the future demand of CPU resource according to the prediction of historical resource utilization and make better use of the shared resource in cloud clusters. Moreover, CPU resource can also be extended to other performance indicators such as memory usage and network bandwidth, etc. The input data dimension for RPTCN can also be scaled according to the number of collected performance indicators. Therefore, RPTCN can serve cluster resource management better as a predictor to allocate suitable resource.

Inevitably, RPTCN has some defects that we haven't solved. The accuracy of the model has a certain improvement, but the improvement is not so obvious. We will optimize it in future research. For example, in the dimension expansion, we should consider the value of correlation between CPU resource and other performance indicators, and set different dimension columns according to the correlation weights of each performance metric with predicted resource, so as to improve the prediction accuracy and reduce the training time cost. Moreover, the current resource dimension expansion method is relatively simple. It only adds the value of historical data to the input data columns. In the future, the model expansion method can be improved, such as adding first-order difference information for resource utilization, to further improve the accuracy of the model. In addition, to explore the influence

of TCNs parameters on the running time of this model, we can improve the model training and prediction speed from the perspective of the model parameters, and further apply the model to the real-time resource usage prediction.

VI. Related Work

In this section, we summarize and analyze existing workload prediction models and compare them with the novel prediction model proposed in this paper. From the perspective of model, they are mainly divided into regression-based, machine learning-based, and hybrid-based models. These models are optimized from data processing and model structure to build predictive components that are suitable for specific cases.

A. Regression-based Models

Resource prediction in cloud clusters has a rich history, with innovation still occurring as new models address new workload characteristics and larger scale. The most classic regression model used for time series prediction is ARIMA, the research on ARIMA-based workload prediction frameworks such as [27], [30]–[33], they are used to predict resource usage, virtual machine scaling, and job scheduling, etc.

In [34], Yang et al. present a linear regression model to estimate the workload and apply it in an auto-scaling mechanism to scale virtual resources in real-time scaling and pre-scaling. They consider the pre-scaling using integer programming and introduce a greedy method for accurately predicting which incurs a lower cost. Ray et al. [35] develop a model-predictive algorithm based on ARMA (Autoregressive Moving Average) for resource auto-scaling, experimental results are provided that demonstrate that resources can be allocated and deallocated by this algorithm in a way that satisfies both the application QoS while keeping resource cost low. Doulamis et al. [36] propose an efficient non-linear workload prediction mechanism incorporated with a fair scheduling algorithm for task allocation and resource management in grid computing.

B. Machine Learning-based Models

The rise of machine learning has provided new methods and opportunities for workload prediction. More and more research has been done on the methods of constructing workload prediction models of machine learning.

For instance, in [37], Yu et al. develop a clustering and learning-based approach to predict future resource utilization. They group the workloads into multiple clusters, and then they use neural network to learn the characteristics of each type workload. For each new task, they collect its initial logs, determine it belongs to which cluster, and use the trained neural network of its cluster to predict the future resource usage. Song et al. [38] apply LSTM to predict the mean workload over consecutive future time intervals and actual workload multi-step-ahead, and they use two real-world workload traces to evaluate the performance. One is the workload trace in Google data center, and the other is that in a traditional distributed system. In [39], Yang et al. propose a new method for host workload prediction, which uses an autoencoder as the pre-recurrent feature layer of the echo state networks. Kumar et al. [40] present a workload prediction model using neural network and self-adaptive differential evolution algorithm. The model is capable of learning the best suitable mutation strategy along with optimal crossover rate. Gupta et al. [41] use multivariate LSTM models for the prediction of resource usage for workloads in clouds. They analyze and compare the prediction results of LSTM model and bidirectional LSTM model with fractional difference based methods.

C. Hybrid-based Models

In order to be suitable for different prediction scenarios, many researchers have integrated multiple models to construct workload prediction methods.

In [42], a hybrid methodology that combines ARIMA and ANN (Artificial Neural Network) models is proposed, and it takes advantages of the unique strength of ARIMA and ANN models in linear and nonlinear modeling. Cetinsk et al. [43] propose an advanced model (AME-WPC) for efficient workload prediction in the cloud, which combines statistical and learning methods. They address the workload prediction problem with classification as well as regression and test the solution with random forest on both basic and extended training data. Kumar et al. [44] propose LSTM-RNN. Janardhanan et al. [45] focus on predicting CPU utilization of machines in data centers using LSTM and evaluating it against the widely used and traditional ARIMA models for predicting. Liu et al. [46] propose an elastic resource management method based on workload predicting in edge clouds, this method includes a resource prediction method based on error correction and combines ARMA and ENN (Elman Neural Network) models and a workload migration for minimizing migration time.

D. Compared with RPTCN

Compared with our method RPTCN, these models have two limitations as follows. Firstly, they are almost using one-to-one mode to predict some resource usage, however, the performance logs we collect from systems proved that one kind of performance indicator is related to others strongly, and if we use these indicators as input, it can improve the prediction accuracy. Secondly, we expand the features horizontally in time dimension creatively. This method changes the practice of extending the time length of these models in the vertical direction and has a good performance for high-dynamic resource prediction for a long time.

VII. Conclusion

In this paper, we propose a novel deep learning model for resource usage prediction in the cloud computing environment, which we call *RPTCN*. We add a fully connected layer and attention mechanism on TCNs to predict the resource usage of workloads and use correlation analysis to screen the performance indicators with considering the dependence and correlation among multiple resources. In order to fully describe the relationships of resource usage in the temporal dimension,

we expand the input feature dimensions horizontally in temporal dimension. After verifying and evaluating the method on Alibaba trace v2018, the experimental results have shown that *RPTCN* is more effective (It improves the overall MAE by 6.50%∼89.03% and MSE by 0.41%∼68.82% respectively) for predicting resource usage in cloud clusters comparing with the baselines.

ACKNOWLEDGMENT

This work is supported by Key-Area Research and Development Program of Guangdong Province (NO.2020B010164003), National Natural Science Foundation of China (No. 62072451), Shenzhen Basic Research Program (No. JCYJ20200109115418592), Science and Technology Development Fund of Macao S.A.R (FDCT) under number 0015/2019/AKP, Youth Innovation Promotion Association CAS (NO. 2019349), and Alibaba Innovative Research (AIR) Project.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] T. Dillon, C. Wu, and E. Chang, "Cloud computing: issues and challenges," in *2010 24th IEEE international conference on advanced information networking and applications*. Ieee, 2010, pp. 27–33.

[3] K. S. Rao and P. S. Thilagam, "Heuristics based server consolidation with residual resource defragmentation in cloud data centers," *Future Generation Computer Systems*, vol. 50, pp. 87–98, 2015.

[4] F. Caglar and A. Gokhale, "ioverbook: intelligent resource-overbooking to support soft real-time applications in the cloud," in *2014 IEEE 7th International Conference on Cloud Computing*. IEEE, 2014, pp. 538–545.

[5] C. Lu, K. Ye, G. Xu, C.-Z. Xu, and T. Bai, "Imbalance in the cloud: An analysis on alibaba cluster trace," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 2884–2892.

[6] W. Chen, K. Ye, Y. Wang, G. Xu, and C.-Z. Xu, "How does the workload look like in production cloud? analysis and clustering of workloads on alibaba cluster trace," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2018, pp. 102–109.

[7] Alibaba trace v2018. [Online]. Available: https://github.com/alibaba/clusterdata

[8] F. J. Baldan, S. Ramirez-Gallego, C. Bergmeir, F. Herrera, and J. M. Benitez, "A forecasting methodology for workload forecasting in cloud systems," *IEEE Transactions on Cloud Computing*, vol. 6, no. 4, pp. 929–941, 2016.

[9] C. Liu, C. Liu, Y. Shang, S. Chen, B. Cheng, and J. Chen, "An adaptive prediction approach based on workload pattern discrimination in the cloud," *Journal of Network and Computer Applications*, vol. 80, pp. 35–44, 2017.

[10] B. Fei, X. Zhu, D. Liu, C. Jun-jie, W. Bao, and L. Liu, "Elastic resource provisioning using data clustering in cloud service platform," *IEEE Transactions on Services Computing*, pp. 1–1, 2020.

[11] J. Bi, S. Li, H. Yuan, and M. Zhou, "Integrated deep learning method for workload and resource prediction in cloud systems," *Neurocomputing*, 2020.

[12] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, 2015.

[13] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth *et al.*, "Apache hadoop yarn: Yet another resource negotiator," in *Proceedings of the 4th annual Symposium on Cloud Computing*, 2013, pp. 1–16.

[14] E. Boutin, J. Ekanayake, W. Lin, B. Shi, J. Zhou, Z. Qian, M. Wu, and L. Zhou, "Apollo: Scalable and coordinated scheduling for cloud-scale computing," in *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, 2014, pp. 285–300.

[15] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. H. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center." in *NSDI*, vol. 11, no. 2011, 2011, pp. 22–22.

[16] Z. Zhang, C. Li, Y. Tao, R. Yang, H. Tang, and J. Xu, "Fuxi: a fault-tolerant resource management and job scheduling system at internet scale," in *Proceedings of the VLDB Endowment*, vol. 7, no. 13. VLDB Endowment Inc., 2014, pp. 1393–1404.

[17] A. Tumanov, T. Zhu, J. W. Park, M. A. Kozuch, M. Harchol-Balter, and G. R. Ganger, "Tetrisched: global rescheduling with adaptive plan-ahead in dynamic heterogeneous clusters," in *Proceedings of the Eleventh European Conference on Computer Systems*, 2016, pp. 1–16.

[18] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at google with borg," in *Proceedings of the Tenth European Conference on Computer Systems*, 2015, pp. 1–17.

[19] W. Y. Chen, K. J. Ye, C. Z. Lu, D. D. Zhou, and C. Z. Xu, "Interference analysis of co-located container workloads: A perspective from hardware performance counters," *Journal of Computer science and Technology*, vol. 35, no. 2, pp. 412–417, 2020.

[20] C. Lu, W. Chen, K. Ye, and C.-Z. Xu, "Understanding the workload characteristics in alibaba: A view from directed acyclic graph analysis," in *2020 International Conference on High Performance Big Data and Intelligent Systems (HPBD&IS)*. IEEE, 2020, pp. 1–8.

[21] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[22] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[23] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[25] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[26] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[27] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using arima model and its impact on cloud applications' qos," *IEEE Transactions on Cloud Computing*, vol. 3, no. 4, pp. 449–458, 2014.

[28] N. Tran, T. Nguyen, B. M. Nguyen, and G. Nguyen, "A multivariate fuzzy time series resource forecast model for clouds using lstm and data correlation analysis," *Procedia Computer Science*, vol. 126, pp. 636–645, 2018.

[29] S. Ouhame, Y. Hadi, and A. Ullah, "An efficient forecasting approach for resource utilization in cloud data center using cnn-lstm model," *Neural Computing and Applications*, pp. 1–13, 2021.

[30] D. Yang, P. Jirutitijaroen, and W. M. Walsh, "Hourly solar irradiance time series forecasting using cloud cover index," *Solar Energy*, vol. 86, no. 12, pp. 3531–3543, 2012.

[31] S. Li, Y. Wang, X. Qiu, D. Wang, and L. Wang, "A workload prediction-based multi-vm provisioning mechanism in cloud computing," in *2013 15th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 2013, pp. 1–6.

[32] Y. Wang, C. Wang, C. Shi, and B. Xiao, "Short-term cloud coverage prediction using the arima time series model," *Remote Sensing Letters*, vol. 9, no. 3, pp. 274–283, 2018.

[33] J. Bi, L. Zhang, H. Yuan, and M. Zhou, "Hybrid task prediction based on wavelet decomposition and arima model in cloud data center," in *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*. IEEE, 2018, pp. 1–6.

[34] J. Yang, C. Liu, Y. Shang, Z. Mao, and J. Chen, "Workload predicting-based automatic scaling in service clouds," in *2013 IEEE Sixth International Conference on Cloud Computing*. IEEE, 2013, pp. 810–815.

[35] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *2011 IEEE 4th International Conference on Cloud Computing*. IEEE, 2011, pp. 500–507.

[36] N. Doulamis, A. Doulamis, A. Litke, A. Panagakis, T. Varvarigou, and E. Varvarigos, "Adjusted fair scheduling and non-linear workload prediction for qos guarantees in grid computing," *Computer Communications*, vol. 30, no. 3, pp. 499–515, 2007.
[37] Y. Yu, V. Jindal, I.-L. Yen, and F. Bastani, "Integrating clustering and learning for improved workload prediction in the cloud," in *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*. IEEE, 2016, pp. 876–879.
[38] B. Song, Y. Yu, Y. Zhou, Z. Wang, and S. Du, "Host load prediction with long short-term memory in cloud computing," *The Journal of Supercomputing*, vol. 74, no. 12, pp. 6554–6568, 2018.
[39] Q. Yang, Y. Zhou, Y. Yu, J. Yuan, X. Xing, and S. Du, "Multi-step-ahead host load prediction using autoencoder and echo state networks in cloud computing," *The Journal of Supercomputing*, vol. 71, no. 8, pp. 3037–3053, 2015.
[40] J. Kumar and A. K. Singh, "Workload prediction in cloud using artificial neural network and adaptive differential evolution," *Future Generation Computer Systems*, vol. 81, pp. 41–52, 2018.
[41] S. Gupta and D. A. Dinesh, "Resource usage prediction of cloud workloads using deep bidirectional long short term memory networks," in *2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. IEEE, 2017, pp. 1–6.
[42] G. P. Zhang, "Time series forecasting using a hybrid arima and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, 2003.
[43] K. Cetinski and M. B. Juric, "Ame-wpc: Advanced model for efficient workload prediction in the cloud," *Journal of Network and Computer Applications*, vol. 55, pp. 191–201, 2015.
[44] J. Kumar, R. Goomer, and A. K. Singh, "Long short term memory recurrent neural network (lstm-rnn) based workload forecasting model for cloud datacenters," *Procedia Computer Science*, vol. 125, pp. 676–682, 2018.
[45] D. Janardhanan and E. Barrett, "Cpu workload forecasting of machines in data centers using lstm recurrent neural networks and arima models," in *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE, 2017, pp. 55–60.
[46] B. Liu, J. Guo, C. Li, and Y. Luo, "Workload forecasting based elastic resource management in edge cloud," *Computers & Industrial Engineering*, vol. 139, p. 106136, 2020.