Towards Accurate Prediction for High-Dimensional and Highly-Variable Cloud Workloads with Deep Learning

Zheyi Chen, Jia Hu*, Geyong Min*, Albert Y. Zomaya, *Fellow, IEEE*, and Tarek El-Ghazawi, *Fellow, IEEE*

Abstract—Resource provisioning for cloud computing necessitates the adaptive and accurate prediction of cloud workloads. However, the existing methods cannot effectively predict the high-dimensional and highly-variable cloud workloads. This results in resource wasting and inability to satisfy service level agreements (SLAs). Since recurrent neural network (RNN) is naturally suitable for sequential data analysis, it has been recently used to tackle the problem of workload prediction. However, RNN often performs poorly on learning long-term memory dependencies, and thus cannot make the accurate prediction of workloads. To address these important challenges, we propose a deep Learning based Prediction Algorithm for cloud Workloads (L-PAW). First, a top-sparse auto-encoder (TSA) is designed to effectively extract the essential representations of workloads from the original high-dimensional workload data. Next, we integrate TSA and gated recurrent unit (GRU) block into RNN to achieve the adaptive and accurate prediction for highly-variable workloads. Using real-world workload traces from Google and Alibaba cloud data centers and the DUX-based cluster, extensive experiments are conducted to demonstrate the effectiveness and adaptability of the L-PAW for different types of workloads with various prediction lengths. Moreover, the performance results show that the L-PAW achieves superior prediction accuracy compared to the classic RNN-based and other workload prediction methods for high-dimensional and highly-variable real-world cloud workloads.

Index Terms—Cloud computing, workload prediction, resource provisioning, sequential data analysis, deep learning

1 Introduction

As one of the most prevailing computing paradigms [1], cloud computing promises on-demand provisioning of computing, storage and networking resources with service level agreements (SLAs) between cloud service providers (CSPs) and users. When user requests arrive simultaneously, workloads burst so the available resources might be insufficient. On the contrary, the idle status occurs when workloads stay at a low level, resulting in resource waste. Workload variations lead to the over-provisioning or under-provisioning of resources, which causes unnecessary overheads or poor SLAs [2], [3]. Therefore, CSPs must be able to rapidly determine the strategies of resource provisioning for guaranteeing SLAs while improving resource utilization [2]. To achieve these objectives, the fast and adaptive methods for workload prediction are necessary for cloud computing [4]. Based on the effective prediction for future workloads, more efficient and rational resource provisioning can be achieved through configuring and allocating resources in advance. However, workload prediction in cloud computing faces two main challenges as follows:

- **High variance of workload patterns**. Workload patterns (e.g., the randomly-changing workloads in Google cloud data centers [5], and the autocorrelated and periodic workloads in the DUX-based cluster [6]) vary on different time scales (e.g., seconds and days) in cloud computing. According to the analysis report of Alibaba cloud data centers [7], the average CPU utilization of their entire cluster changes from 5% to 80% during a day with high fluctuations. This high variance of workload patterns makes it very hard to accurately and effectively predict workloads.
- **High dimensionality of workload data**. Workload data in cloud computing usually suffers from the problem of high-dimensional space [8]. For instance, a 1000-dimensional set of workload data is required to be constructed and used as the input of a prediction model for training purposes when there are 1,000 working machines in a cloud data center. This high-dimensional data with redundant and noisy information not only results in more errors in workload prediction but also leads to higher computational overheads of a prediction model [9].

In response to the high variance of workload patterns, the correlations of workload patterns should be effectively captured and utilized in order to develop an accurate workload prediction method that can adapt well to highly-variable workloads. To tackle the challenge of high dimensionality of workload data, the features of original workload data should be further analyzed and extracted in order to reduce the dimensionality of workload data and prediction errors for more efficient and accurate workload prediction.

The problem of workload prediction has received considerable research interests. However, many classic methods are based on the regression theories, heuristics or tra-

- *Zheyi Chen, Jia Hu, and Geyong Min are with the Department of Computer Science, College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter EX4 4QF, United Kingdom. E-mail: {zc300, j.hu, g.min}@exeter.ac.uk.*
- *Albert Y. Zomaya is with the School of Computer Science, The University of Sydney, Camperdown NSW 2006, Australia. E-mail: albert.zomaya@sydney.edu.au.*
- *Tarek El-Ghazawi is with the Department of Electrical and Computer Engineering, The George Washington University, Washington DC 20052, United States. E-mail: tarek@gwu.edu.*

Manuscript received 14 May. 2019; revised 8 Nov. 2019; accepted 9 Nov. 2019. Date of publication XX XX. 2019; date of current version XX XX. 2019.
**Corresponding authors.*
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. XXXX

ditional neural networks, which require workloads with obvious regularity or clear tendency in order to achieve the accurate workload prediction. Meanwhile, traditional neural networks do not make full use of the correlation between neurons for better prediction results. Thus, they cannot effectively realize the accurate prediction of highly-variable workloads. Furthermore, most of these methods focus on the workloads in a small-scale grid or high-performance computing (HPC) system, which present much lower variance compared to a large-scale cloud computing system (e.g., cloud data centers) [10]. Therefore, these methods cannot well adapt to the real-world cloud computing environment with highly-variable workloads, which may result in the severe degradation in the accuracy of workload prediction.

Due to its excellent capability of sequential processing, a recurrent neural network (RNN) [11] can be used to handle the prediction problem with highly-variable workloads. However, it would be an essential challenge to train an effective RNN for workload prediction. Due to the problem of gradient vanishing, the traditional RNN cannot efficiently learn long-term memory dependencies. Some variant RNNs, such as long short-term memory (LSTM) [11] and gated recurrent unit (GRU) [12], exhibit a powerful ability to address this issue [13]. Especially, compared to LSTM, GRU can achieve not only comparable prediction accuracy but also higher learning efficiency with fewer parameters.

However, due to the high dimensionality of workload data, training an RNN-based prediction model is a time-consuming task with high computational complexity. To address this issue, one way is to reduce the dimensionality of workload data by extracting the essential feature representations of the original workload data. Some methods have been proved to be good candidates for reducing the data dimensionality such as the principal component analysis (PCA) [14] and auto-encoder [11]. However, PCA relies on linear methods to find the direction of the largest variance in high-dimensional data, which limits the types of dimension that can be reduced. By contrast, the auto-encoder based methods (e.g., sparse auto-encoder [11]) overcome this limitation by introducing the nonlinearities of neural networks. But the classic auto-encoder usually overuses hidden units, which results in low efficiency when reducing the dimensionality of workload data.

To solve these essential challenges in workload prediction, we first design a top-sparse auto-encoder (TSA) to efficiently reduce the dimension of workload data. Next, the compressed workloads are used as the input of the proposed deep Learning based Prediction Algorithm for cloud Workloads (L-PAW), in order to achieve an adaptive, precise and efficient prediction for highly-variable workloads in cloud computing. The main contributions of this paper are summarized as follows:

- A top-sparse auto-encoder (TSA) is designed to efficiently extract the low-dimensional but essential feature representations of workloads from the original high-dimensional workload data, which is able to achieve the effective workload compression through selecting the hidden units with high-level activation degrees.
- An efficient deep Learning based Prediction Algorithm for cloud Workloads (L-PAW) is proposed to learn long-term memory dependencies from historical workloads through integrating the TSA and GRU block into RNN. The L-PAW can adapt well to highly variable workloads and obtain the accurate workload prediction through capturing the essential historical information with the settings of update and reset gates in the GRU block.
- Extensive simulation experiments using the real-world workload traces are conducted to validate the effectiveness and adaptability of the proposed L-PAW on cloud workload prediction. The results demonstrate that the L-PAW outperforms the classic RNN-based and other prediction methods for high-dimensional and highly-variable real-world cloud workloads.

The rest of this paper is organized as follows. Section 2 introduces the preliminaries of SA and RNN. In Section 3, we analyze the related work on workload prediction. Section 4 describes the system model. In Section 5, the proposed TSA and L-PAW are discussed in detail. Section 6 evaluates the proposed methods via simulation experiments with real-world datasets of cloud workloads. Finally, we conclude this paper in Section 7.

2 PRELIMINARIES

In this section, the sparse auto-encoder (SA) and recurrent neural network (RNN) are briefly introduced. Based on the basic principles of these two techniques, we aim to solve the problem of high dimensionality and high variance on cloud workload prediction.

2.1 Sparse Auto-Encoder

Sparse auto-encoder (SA) [11] can automatically learn the essential feature representations from the unlabeled data. In practice, the representations extracted by SA can be used to replace the original data, which often leads to better performance for the learning process of neural networks. More specifically, SA is constructed with a single hidden layer, which requires the output data to be as similar as possible to the input data. Moreover, the hidden layer must satisfy certain sparsity, which also means that the hidden layer cannot carry too much information. Therefore, the input data would be compressed in the hidden layer and decompressed in the output layer.

SA learns a function $y_i = f(Wx_i + b) \approx x_i$, where x_i and $y_i \in \mathcal{R}^n$. In other words, SA targets to approximate an identity function so that the output y_i can be close to the input x_i. During this process, some important features would be extracted from the input data. Moreover, the input data can be replaced by hidden units (i.e., neurons of the hidden layer), which achieves the effect of data compression. Even if the number of hidden units is large, the essential features of the input data can still be found by adding the sparsity constraint. Especially, $a_j^{(h)}$ is used to represent the activation degree of the hidden unit j and its average, $\hat{\rho}_j$, can be denoted as follows:

$$\hat{\rho}_j = \frac{1}{n} \sum_{i=1}^{n} \left[a_j^{(h)}(x_i) \right]. \tag{1}$$

$\hat{\rho}_j$ is commonly enforced to be ρ, where ρ is the sparsity parameter and should be close to 0 for satisfying the sparsity

constraint on hidden units. Therefore, the following term is used as a penalty when $\hat{\rho}_j$ seriously deviates from ρ.

$$\sum_{j=1}^{N_h} \rho \ln \frac{\rho}{\hat{\rho}_j} + (1-\rho) \ln \frac{1-\rho}{1-\hat{\rho}_j}, \tag{2}$$

where N_h is the number of hidden units and the above formula can also be denoted as $\sum_{j=1}^{N_h} KL(\rho||\hat{\rho}_j)$ according to the Kullback-Leibler (KL) divergence [11].

As a standard function for measuring the level of difference between two particular distributions, the minimum of KL-divergence can be reached when $\hat{\rho}_j$ is close to ρ, which also means that the process for minimizing the penalty term has the same effect on approximating $\hat{\rho}_j$ to ρ. Thus, the overall cost function of SA, $J_{sparse}(W, b)$, is as follows:

$$J_{sparse}(W, b) = J(W, b) + \beta \sum_{j=1}^{N_h} KL(\rho||\hat{\rho}_j), \tag{3}$$

where $J(W, b)$ is the cost function of neural networks and β is the weight used to control the sparsity penalty term.

2.2 Recurrent Neural Network

Recurrent neural network (RNN) [11] emphasizes the connectivity of neurons between hidden layers, which can be used to process the sequential problems through using historical memories. Typically, the hidden layers of a traditional neural network are fully-connected or partially-connected, but the neurons between different neural networks are connectionless. By contrast, RNN aims to use a sequence to construct a relationship between historical memories and the current status. Therefore, the neurons between hidden layers in RNN are connected, which also means that the input of hidden layers not only comes from the input layer at the current moment but also the output of hidden layers at the previous time. The structure of classic RNN is shown in Fig. 1, where the chained features reveal that RNN is essentially related to sequential processing.

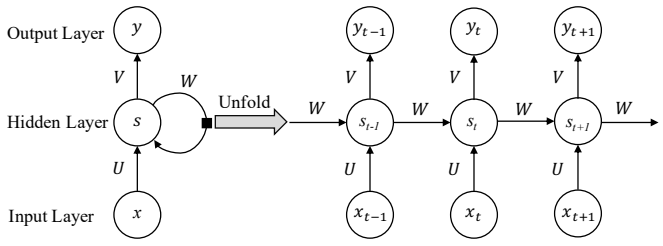


Fig. 1. The structure of classic RNN.

Based on the historical memories and the current input, the future output can be predicted as follows:

$$s_t = \tanh(U \cdot x_t + W \cdot s_{t-1}), y_t = softmax(V \cdot s_t), \tag{4}$$

where s_t, x_t, and y_t denote the hidden status, the input and the output at time t, respectively.

3 Related Work

Workload prediction in cloud computing has attracted much research attention, while many scholars have contributed to addressing this important problem. In this section, we first review the classic methods and then present the RNN-based approaches for cloud workload prediction.

3.1 Classic Methods for Workload Prediction

An auto-regression (AR) based prediction model was designed in [15], where the time-series based historical CPU utilization was used to predict the future workloads. However, this model is strictly linear in essence and lack the adaptability to highly-variable workloads in a more complex cloud environment. Linear regression (LR) and wavelet neural network (WNN) were integrated for the short-term prediction of workloads in [16], but it suffers from high errors for the long-term prediction. Kumar and Singh [17] proposed a workload prediction model by combining artificial neural network (ANN) and self-adaptive differential evolution algorithm, which promises higher prediction accuracy than LR and the back-propagation based methods. A clustering-based learning approach was presented in [18] for enhancing the accuracy of workload prediction, but it is hard to determine a proper learning rate. The authors proposed two collaborative filtering techniques using k-nearest neighbors (k-NN) to obtain the good performance of workload prediction among various multi-core systems [19]. But k-NN is inefficient, which usually leads to huge computational overheads. Swarm and evolutionary optimization algorithms are applied to train the neural networks for predicting the host utilization [20], but this approach might suffer from the difficulty of selecting parameters (e.g., mutation and crossover rates). Kaur *et al.* [21] developed an ensemble-based prediction method for CPU usage of scientific applications, which takes the average accuracy of eight regression-based prediction models into the consideration of final prediction results. However, this method might be restricted by the long training time for different models.

In general, most of the classic methods for workload prediction depend on the heuristics, traditional neural networks or regression-based methods. Therefore, they require workloads with obvious regularity or clear tendency in order to achieve precise prediction. For instance, without utilizing the correlations between neurons, the traditional neural networks cannot achieve the accurate workload prediction effectively. Moreover, these methods mainly execute workload prediction in a small-scale grid or HPC systems, which exhibit lower variance compared to the large-scale cloud data centers. To effectively solve these important challenges and achieve better prediction performance for highly-variable workloads in the cloud environment, more intelligent strategies are demanded. The emergence of RNN, a competent architecture for sequence processing, presents great potentials for cloud workload prediction.

3.2 RNN-Based Approaches for Workload Prediction

RNN [11] emphasizes the connectivity of neurons between hidden layers in order to efficiently process the sequential problem through using the historical memories in neural

networks. Over the past few years, RNN has also been used to deal with the problem of workload prediction in cloud computing. Zhang *et al*. [22] proposed an RNN-based model for improving the accuracy of workload prediction. Similarly, the classic RNN architecture was adopted in [23] and [24] to forecast the future workloads in cloud data centers. It turns out that RNN can work well when coping with short-term dependencies. However, the authors in [25] and [26] have proved that RNN cannot effectively guarantee the excellent performance of long-term prediction. This is because that the traditional RNN is unable to address the problem of gradient vanishing in training. Therefore, RNN will lose the ability to connect and use the meaningful information when the distance between the information and the predicted value increases. This problem is known as the long-term dependencies. To solve this problem, LSTM was developed as an improvement of RNN [11], in order to better deal with long-term dependencies. Song *et al*. [27] utilized LSTM for the host load prediction, which improved their previous work using RNN-based echo state networks (ESN) [28]. Similarly, an LSTM-based model using association learning was proposed to capture the relationship among different resource metrics to achieve accurate prediction for future workloads [29]. Compared with LSTM, GRU is able to converge more easily with fewer settings of parameters [12]. But there is little research work [30] using GRU-based approaches with the consideration of training efficiency of neural networks for workload prediction in the cloud environment.

Overall, most of the RNN-based approaches depend on the classic RNN architecture, which can neither survive the problem of gradient vanishing nor capture long-term memory dependencies from historical workloads. Therefore, they are unable to learn the precise prediction results for cloud workloads with high variance or adapt to the real-world requirements of cloud computing, which might result in the deterioration of prediction accuracy and efficiency. Although there exist small amount of research using improved forms of RNN (e.g., LSTM and GRU) to address the issue of gradient vanishing [13], the problem of high-dimensional workload data in cloud computing has not been well considered. This might lead to low accuracy and high computational complexity for workload prediction.

In order to address these open challenges, we first propose TSA to improve the compression efficiency in response to the high dimensionality of workload data. Next, to better deal with the high variance of workload patterns, the TSA and GRU block are integrated into an RNN structure to capture long-term memory dependencies in neural networks for achieving the accurate prediction of future workloads.

4 System Model

CSPs promise the rapid resource provisioning to meet user requests, which enables the scale of servers to be expanded or reduced based on the current resource usage so that the better load balancing can be achieved in a cloud data center. However, due to workload variations, it is impossible to determine an ideal resource provisioning strategy through immediate operations, which dramatically degrades the user experience. Meanwhile, inefficient and unreasonable resource provisioning would also lead to unnecessary overheads (e.g., energy consumption) or SLAs violations. In response to the high variance and high dimensionality of workloads, we propose a workload prediction model for cloud data centers in order to minimize the errors between the predicted workloads and the actual ones while maintaining the satisfactory processing efficiency. The key components of the proposed model are shown in Fig. 2.

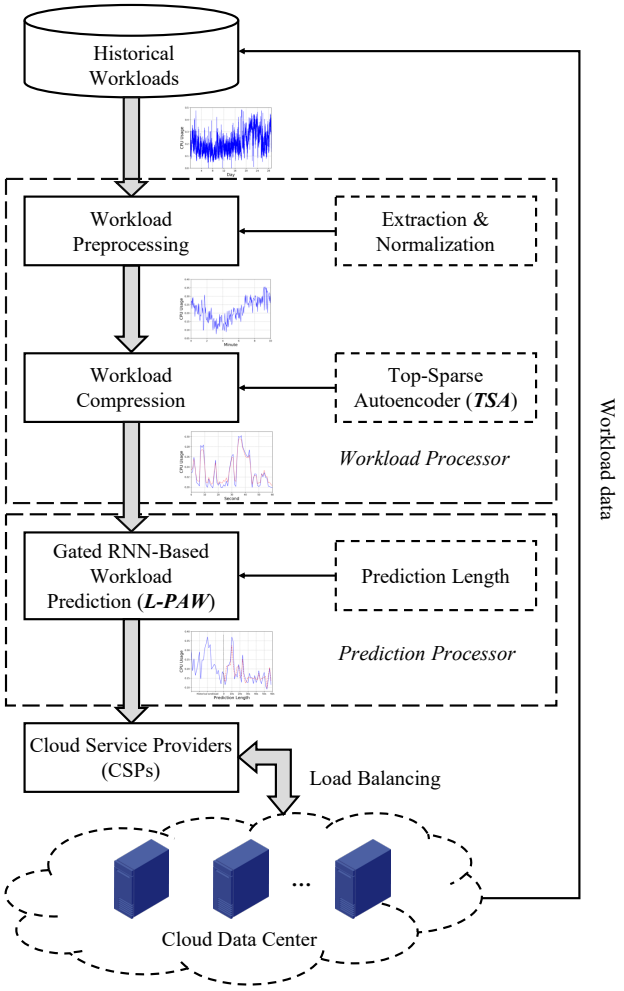


Fig. 2. The proposed workload prediction model in cloud data center.

Workload processor: Historical workload data from the cloud data center is used in the proposed prediction model. After the preprocessing and compression of historical workloads, workload data is regarded as the input of the prediction processor. Generally, the historical workload data includes various metrics about the system running status (e.g., CPU usage, memory usage, and disk I/O time), which increases the redundancy and complexity to the computation process. According to the published data from Google's production cluster [5] and Amazon AWS EC2 [31], the average CPU utilization was quite low, at only 20% and 7%, respectively. The huge cost of investment in cloud data centers with low CPU utilization has been a great concern to cloud service providers. Therefore, the industry has seen the CPU utilization as a key factor to improve the resource provisioning in cloud data centers. As many state-of-the-art works [32], [33], [34] on the resource provisioning of cloud

data centers, we also consider CPU utilization as the main performance indicator of workloads and extract this metric during the workload preprocessing, which is denoted as $\vec{X} = (x_1, x_2, ..., x_n)$, where $n \in \mathcal{R}$ and x_n is the CPU usage at time n. As there exists a huge difference in the value range of workload data in different time intervals, the original workload data needs to be normalized before proceeding to the next step, which can help accelerate the convergence of learning-based algorithms. More specifically, we adopt one of the most widely used normalization methods in machine learning (i.e., standardization) as follows:

$$x' = \frac{x - mean(\vec{X})}{\sigma}, \tag{5}$$

where $mean(\vec{X})$ is the mean value of $\vec{X}$ and $\sigma = \sqrt{E(\vec{X}^2) - (E(\vec{X}))^2}$ is the standard deviation.

After preprocessing, the normalized workload data $\vec{X}'$ is forwarded to the workload compression. The high dimensionality and redundancy of the workload data can seriously degrade the prediction accuracy and lead to high computational complexity. To this end, a top-sparse auto-encoder (TSA) is proposed to compress the workload data, which effectively extracts lower-dimensional but essential feature representations of workload data as the input of the gated RNN-based workload prediction in the next step. The detailed description of the TSA is given in Section 5.

Prediction processor: Using the normalized and compressed historical workload data from workload processor, the future workloads are predicted by prediction processor and transferred to CSPs, who will use the predictions to determine the suitable resource provisioning strategies for load balancing in a cloud data center. In the prediction processor, the L-PAW, a gated RNN-based learning method, is proposed for capturing long-term memory dependencies from historical workloads in order to achieve more accurate prediction for the problem of time series. Before adopting the L-PAW for workload prediction, the CPU usage of each trace measured during each time interval is added into the historical workloads and used as the input of RNN. Through the setting of the time length of prediction, the workload prediction for different future periods can be realized. The details of the L-PAW are given in Section 5. We then use the mean-squared error (MSE) [11] to measure the accuracy of workload prediction:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2, \tag{6}$$

where N denotes the time length of prediction, $\hat{y}_i$ and y_i are the predicted workload and the actual one, respectively.

5 Deep Learning based Prediction for Cloud Workloads

This section presents the proposed L-PAW, a deep learning based algorithm for cloud workload prediction. First, a top-sparse auto-encoder (TSA) is designed to efficiently extract the low-dimensional but essential feature representations of the workload data. Second, the TSA and GRU block are integrated into RNN to capture long-term memory dependencies from historical workloads for achieving the efficient and precise workload prediction.

As shown in Fig. 3, the input of the TSA is a vector of workload examples $\vec{X} = (x_1, x_2, ..., x_n)$, where $n \in \mathcal{R}$ and x_n is the CPU usage at time n. Similar to SA, TSA also tries to approximate an identity function $y_n = f(Wx_n + b) \approx x_n$ so that the output y_n can be close to the input x_n. Typically, SA is a combination of linear activation functions and fixed weights, which usually leads to the overuse of hidden units and results in low learning efficiency. The proposed TSA can also be regarded as an improved form of SA, where top k hidden units with the highest activation degree are selected for reconstructing the input data rather than using all hidden units as SA does. During the forward propagation, the average activation degree of each hidden unit, $\hat{\rho}$, is calculated by Eq. (7) as follows:

$$\hat{\rho} = \frac{1}{n}\sum_{i=1}^{n}[a^{(h)}(x_i)], \tag{7}$$

where $a^{(h)}$ is the activation function of the hidden layer.

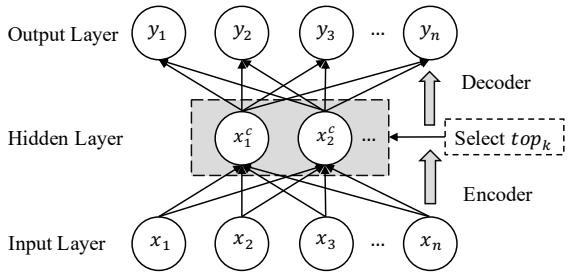


Fig. 3. The network structure of the proposed TSA.

Next, all hidden units are sorted by their respective values of $\hat{\rho}$ and the top k hidden units can be identified, which is denoted as a vector $\tau = top_k(\hat{\rho})$. Therefore, the non-linear calculation only occurs while processing $top_k(\hat{\rho})$, which greatly reduces the computational complexity compared to SA. More specifically, the value of k affects the similarity between workload data before and after compression. For example, the TSA cannot fully capture the features of raw data while using a small value of k (too few hidden units), which will distort the compressed data. Conversely, the TSA may contain much redundant information while using a large value of k (too many hidden units), which will increase the complexity of the following prediction work. The key steps of the proposed TSA are illustrated in Algorithm 1. The complexity of Algorithm 1 is $O(n)$, linear to the size n of the hidden layer in TSA.

Therefore, the problem of workload compression is transformed to the computation for the weight W and the bias b through minimizing the cost function $J_{TSA}(W, b)$. Especially, the cost function of standard neural networks, $J(W, b)$, is shown as follows:

$$J(W,b) = \frac{\lambda}{2n}\sum_{l=1}^{2}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2 + \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{2}\left\|x_i - y_i\right\|^2\right), \tag{8}$$

Algorithm 1: Top-Sparse Auto-encoder (**TSA**)

Input: workload examples $\vec{X} = (x_1, x_2, ..., x_n)$
Initialize: the number of hidden units N_h and the number of hidden units with the largest activation k, where $k < N_h$
for *each training epoch* $n = 1, 2, ..., N$ **do**
  for *each hidden unit* $j = 1, 2, ..., N_h$ **do**
    Execute the forward propagation and compute the average activation degree of hidden units $\hat{\rho}_j = \frac{1}{n}\sum_{i=1}^{n}\left[a_j^{(h)}(x_i)\right]$
  end
  Select top k hidden units from $\hat{\rho}$ with the largest activation and set the others to zero $\hat{\rho}_{(\tau)^c} = 0$, where $\tau = top_k(\hat{\rho})$
  Compute the cost function of TSA $J_{TSA}(W, b) = J(W, b) + \beta\sum_{j=1}^{k} KL(\rho||\hat{\rho}_j)$
  Ouput the compressed workloads $x_n^c = Wx_n + b$
  Execute the backpropagation of cost $J_{TSA}(W, b)$ through the definition of $\tau = top_k(\hat{\rho})$
end

where the first item is the regularization for avoiding overfitting and the second item is the MSE between the original workload x_i and the decoded one y_i.

In order to merge the KL-divergence [11] into the computation for derivative, the original derivative of hidden layers during the backpropagation is modified as Eq. (9). During the forward propagation, all training samples should be calculated for obtaining the average activation degree $\hat{\rho}_i$ before processing to the backpropagation.

$$\delta_i^{(h)} = \left[\sum_{j=1}^{N_o}\delta_j^{(o)}W_{ji}^{(h)} + \beta\left(-\frac{\rho}{\hat{\rho}_i} + \frac{1-\rho}{1-\hat{\rho}_i}\right)\right]f'(z_i^{(h)}), \quad (9)$$

where N_o is the number of output units and $f'(z_i^{(h)})$ is the derivative of activation $f(z_i^{(h)}) = a_i^{(h)}$.

Then, we regard the compressed workloads as the high-level feature representations of the original data and use them as the input vector of the RNN-based workload prediction, which is denoted as $\vec{X}^c = (x_1^c, x_2^c, ..., x_t^c)$. Suppose the vector of predicted workloads is $\vec{Y} = (\hat{y}_1, \hat{y}_2, ..., \hat{y}_t)$, then the prediction model is trained by comparing the errors between the predicted workload $\hat{y}_t$ and the actual one x_{t+1}^c, where x_{t+1}^c represents the actual workload at time $t+1$. Especially, the backpropagation through time (BPTT) [11] is adopted as the training algorithm for RNN. When there exists only a short time interval between the historical workload and the predicted one, RNN can learn useful information for effective prediction. However, RNN reads and updates all previous information, the accumulation of gradients in RNN will be close to 0 as the time interval increases. As a result, network parameters of RNN cannot be updated effectively and RNN gradually fails to learn. This problem is known as gradient vanishing [12], which can also be expressed as the poor ability for capturing long-term memory dependencies. Therefore, the historical workloads from a long time ago cannot be effectively used for workload prediction through traditional RNN structure.

Algorithm 2: Deep Learning based Prediction Algorithm for cloud Workloads (**L-PAW**)

Input: workload examples $\vec{X} = (x_1, x_2, ..., x_t)$
Initialize: the learning rate γ, the learning rate decay λ, the truncated number T, the epoch threshold E_t, and the batch size N_{bs}
Call **TSA** to compress X and obtain the new input $\vec{X}^c = (x_1^c, x_2^c, ..., x_t^c)$
for *each training epoch* $n = 1, 2, ..., N$ **do**
  Segment the epoch N $E_t = segment(N)$
  if $n > E_t$ **then**
    $\gamma = \gamma * \lambda$
  end
  for *each truncated number* $t = 1, 2, ..., T$ **do**
    The updating mode of the update gate z_t $z_t = \sigma(W_z \cdot [\hat{y}_{t-1}, x_t^c])$
    The updating mode of the reset gate r_t $r_t = \sigma(W_r \cdot [\hat{y}_{t-1}, x_t^c])$
    Compute the new memory content $\widetilde{y_t}$ $\widetilde{y_t} = \tanh(W \cdot [r_t \cdot \hat{y}_{t-1}, x_t^c])$
    The output of the GRU block $\hat{y}_t = (1 - z_t) \cdot \hat{y}_{t-1} + z_t \cdot \widetilde{y_t}$
    for $i = 1, 2, ..., N_{bs}$ **do**
      Train $GRU(W_z, W_r, W)$ by using the mini-batch SGD
    end
  end
end

To this end, we propose the L-PAW for better addressing the above problems in workload prediction. Based on the essential feature representations of workloads extracted by the proposed TSA, we replace the hidden layers of classic RNN with GRU blocks. The key steps of the L-PAW are shown in Algorithm 2. After calling the TSA to obtain the compressed workloads, we set a learning rate decay λ to control the learning rate γ segmentally, which aims to achieve more efficient learning at different stages for training the neural networks. To solve the problem of gradient vanishing occurs in the traditional RNN structure, some gated RNNs were proposed, such as LSTM [11] and GRU [12]. Compared to LSTM, GRU can achieve higher learning efficiency with fewer parameters. Different from the traditional RNN, the GRU utilizes gate structures to selectively read and update the previous information. Therefore, the GRU only reserves the information that is useful for prediction and filters out irrelevant information. Meanwhile, the GRU automatically creates short links between different network layers by using gate structures and directly transfers the previous information it reserved. Therefore, the GRU is able to solve the problem of gradient vanishing by re-parametrizing the traditional RNN [13] based on the settings of different gate structures. The core idea of GRU is to make hidden units to preserve some long-term memories, which enables the gradient to be progressed over many timesteps. GRU is a simplified form of LSTM that merges the forget gate and

the input gate of LSTM into an update gate. Thus, GRU consists of two gates, which are the update gate z_t and the reset gate r_t. As shown in Fig. 4, we illustrate the structure of GRU block in the proposed L-PAW algorithm. Similar to LSTM, the update mode of these two gates is based on the current input x_t^c and the previous hidden status $\hat{y}_{t-1}$. The new memory content $\widetilde{y}_t$ is regarded as the new information at current time t, where the reset gate r_t is used to control whether the previous memories need to be retained. Besides, the update gate z_t is used to control whether the previous memory content $\hat{y}_{t-1}$ and the new memory content $\widetilde{y}_t$ is to be forgotten or added. Thus, the output of GRU block $\hat{y}_t$ (the predicted workload) can be calculated based on the update gate z_t. The complexity of Algorithm 2 is related to the model capacity (i.e., the number of parameters in the model), denoted by $O(3(n^2+nm+n))$, where m is the size of input, n is the size of hidden layer, and there are three sets of operations requiring weight matrices in the GRU block (two sets of matrices for update gate and reset gate, and one set of matrices for new memory content). Especially, the GRU is trained by using mini-batch stochastic gradient descent (SGD)for higher accuracy [35].

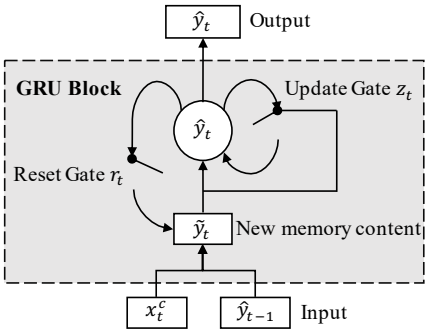
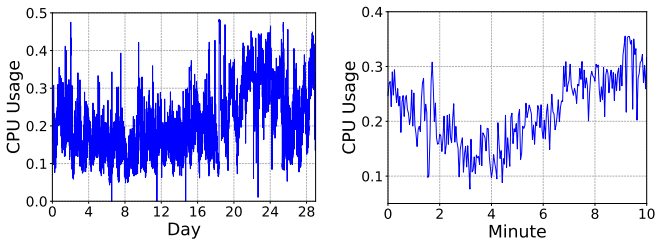


Fig. 4. The structure of GRU block in the L-PAW.

The integration of TSA and GRU block enables the classic RNN to learn long-term memory dependencies from historical workloads more effectively. Whenever historical memories are considered to be critical, the update gate is closed for reserving the essential workload features over multiple time steps. Moreover, the reset gate enables the GRU block to reasonably utilize the model capacity through resetting when the reserved memories are not necessary. Therefore, the proposed L-PAW is built on a simpler structure with fewer gates than LSTM and can also achieve faster convergence speed than GRU with the high-level representations of workload data extracted by the proposed TSA. By contrast, LSTM consists of more gates and parameters, which requires a larger number of training samples and a longer time in order to train a good model. While GRU may encounter the degradation of learning efficiency caused by the overuse of hidden units in the classic SA.

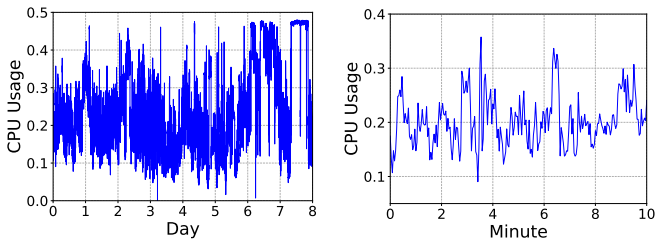
6 Experiments

In this section, we first present the simulation settings and datasets of our experiments. Then, we evaluate the performance of the proposed L-PAW and conduct comparisons with the RNN-based and other classic methods for workload prediction in cloud data centers.



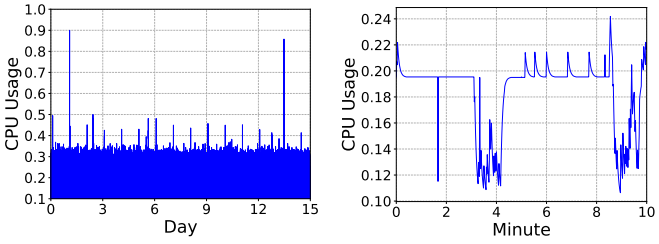
(a) Per-day workloads. (b) Per-minute workloads.

Fig. 5. The highly random workloads in Google cloud data centers.



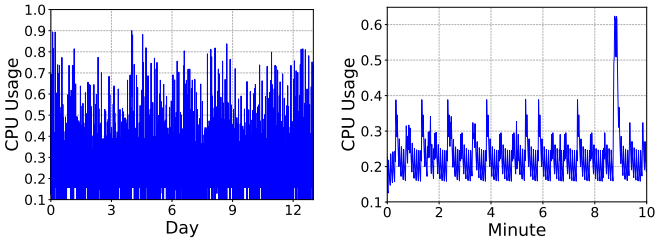
(a) Per-day workloads. (b) Per-minute workloads.

Fig. 6. The highly random workloads in Alibaba cloud data centers.



(a) Per-day workloads. (b) Per-minute workloads.

Fig. 7. The highly autocorrelated workloads in the DUX-based cluster.

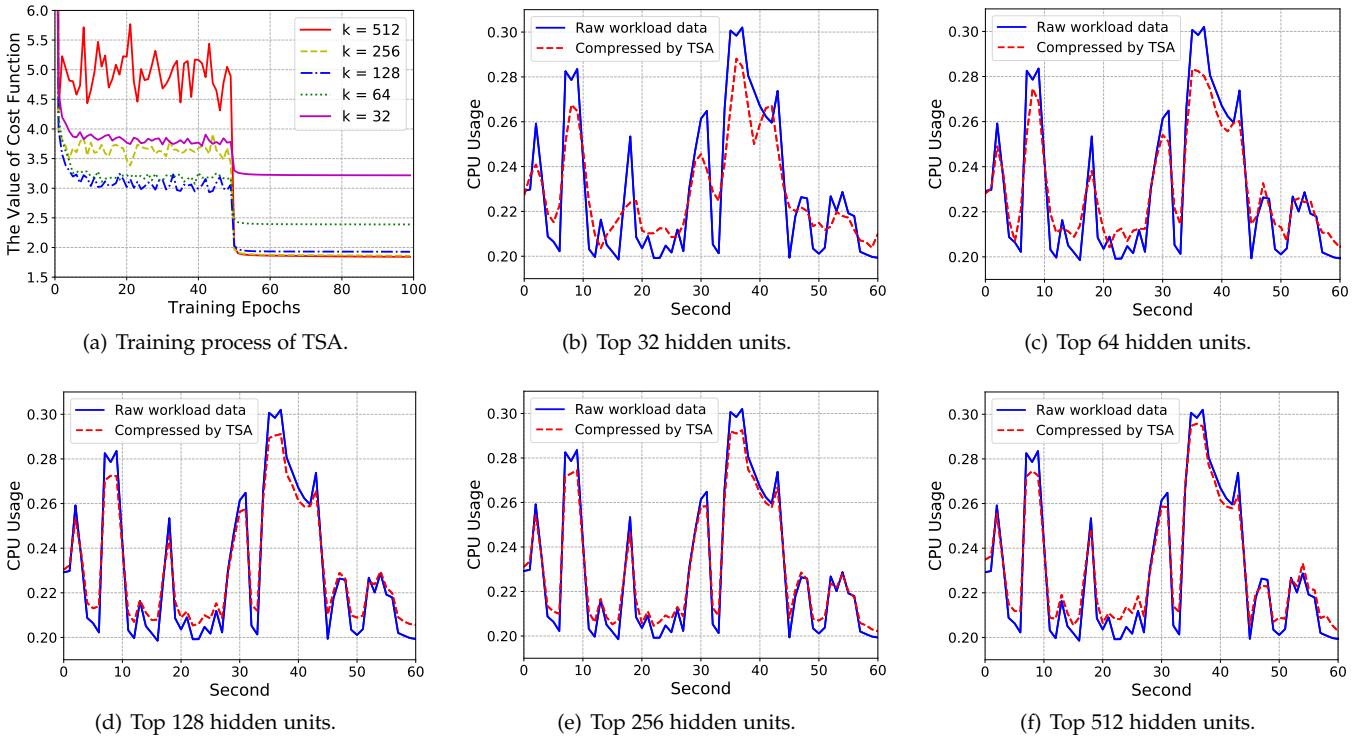


(a) Per-day workloads. (b) Per-minute workloads.

Fig. 8. The highly periodic workloads in the DUX-based cluster.

6.1 Simulation Settings and Datasets

We implement the proposed model for cloud workload prediction based on TensorFlow 1.4.0 [36]. Three real-world datasets are used in the experiments. The first one is Google cluster-usage traces [5], which contain the running information over 125,000 machines in Google cloud data centers during May 2011. The second one is Alibaba cluster traces [7], which include 4,000 machines with the runtime resource usage in 8 days. The third one is the DUX-based cluster traces collected by Dinda [6]. In our experiments, we regard the CPU usage as the main performance index of workloads [32], [33], [34]. More specifically, we randomly select 1,000 machines from Google datasets over 29 days, where each



(a) Training process of TSA. (b) Top 32 hidden units. (c) Top 64 hidden units.

(d) Top 128 hidden units. (e) Top 256 hidden units. (f) Top 512 hidden units.

Fig. 9. Performance evaluation of the TSA for workload data compression with different top k hidden units.

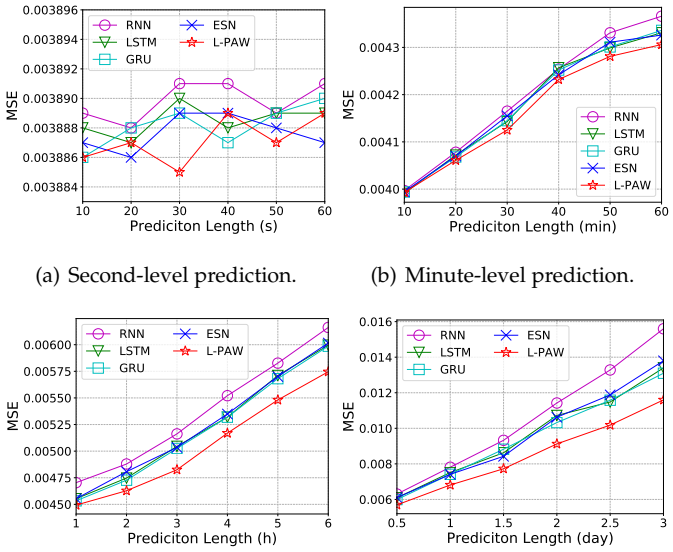
machine contains around 100,000 traces. Similarly, we also randomly select 1,000 machines from Alibaba datasets over 8 days, where each machine contains around 7,000 traces. Then, we extract several essential metrics related to the workload prediction, including the machine ID, start time, end time, CPU usage, memory usage, and disk I/O usage of each trace in both Google and Alibaba datasets. As shown in Figs. 5 and 6, we present the per-day and per-minute workload fluctuation of a machine in Google and Alibaba cloud data centers, respectively. As the DUX-based cluster traces have been classified according to features of workloads, we select two datasets over two specific machines, one contains 1,296,000 highly autocorrelated workload traces over 15 days and another one contains 1,123,200 highly periodic workload traces over 13 days. Figs. 7 and 8 show the per-day and per-minute workload fluctuation of these two machines in the DUX-based cluster, respectively. As we can see from Figs. 5 to 8, the workloads of Google and Alibaba cloud data centers exhibit a more random feature, while the workloads of the DUX-based cluster displays higher autocorrelation and periodicity. In average, the size of workload examples of a host machine is around 8000 after workload preprocessing. We feed data into our prediction model in batches. In more detail, we randomly split the datasets into three parts, which are the training set (50%), the validating set (25%) and the testing set (25%). The training set is used for model training (calculating the weights of neural networks), the validation set is used for model selection (choosing the hyper-parameters and preventing overfitting), and the testing set is used to evaluate the performance of the selected optimal model. Moreover, the total number of training epochs is 100, the initial learning rate is 0.03, the number of truncated backpropagation steps is 32, and the batch size is 128.

6.2 Experimental Results

We first evaluate the performance of the proposed TSA for compressing workloads in terms of the value of cost function and compression effect, using Google datasets with different numbers of top hidden units, where the value of k changes from 32 to 512. Fig. 9(a) illustrates that the value of cost function drops dramatically after around 50 training epochs and gradually converges with different numbers of top hidden units. More specifically, the value of cost is relatively high when the number of top hidden units is small (e.g., $k \leq 64$). Because the neural networks of the TSA have to be reconstructed largely for fitting in the compression requirements when the number of top hidden units is small. When the number of top hidden units is large (e.g., $k \geq 128$), the value of cost function exhibits no obvious distinction among the cases with different numbers of top hidden units, as the network structures are ready to learn the essential feature representations from raw workload data. Thus, we set the number of top hidden units as 128 (i.e., $k = 128$) and use this setting in our following experiments. Meanwhile, we plot the workload data before and after using the TSA under different settings of top hidden units. As shown in Figs. 9(b) to (f), the TSA for workload compression is effective under proper settings of top hidden units, which is able to provide an effective feature representation that can greatly reduce the computational complexity of our proposed method for cloud workload prediction.

Based on the Google datasets and the preprocessed results of workload compression by using the TSA, we evaluate the proposed L-PAW and other recent RNN-based methods for workload prediction, including recurrent neural network (RNN) [22], long short-term memory (LSTM) [27], gated recurrent unit (GRU) [30], and echo state networks

(ESN) [28]. We compare both the prediction accuracy and learning efficiency among these methods, measured by MSE and the average training time, respectively. Fig. 10 shows the MSE of different RNN-based methods with various levels of prediction length. In general, MSE rises with the increase of prediction length for all these methods. More specifically, for the second-level prediction, there is not much difference in prediction accuracy between the L-PAW and other RNN-based methods. With the increase of prediction length (from the minute-level prediction to the day-level prediction), the L-PAW exceeds other RNN-based methods in terms of prediction accuracy and exhibits a bigger gap in performance improvements. This is because that the L-PAW can address the problem of gradient vanishing and capture long-term memory dependencies from historical workloads. The results demonstrate that the L-PAW is more effective for workload prediction than other RNN-based methods under the high-dimensional and highly-variable cloud workloads.



(a) Second-level prediction. (b) Minute-level prediction.

(c) Hour-level prediction. (d) Day-level prediction.

Fig. 10. Prediction accuracy (MSE) of different RNN-based methods with various levels of prediction length.

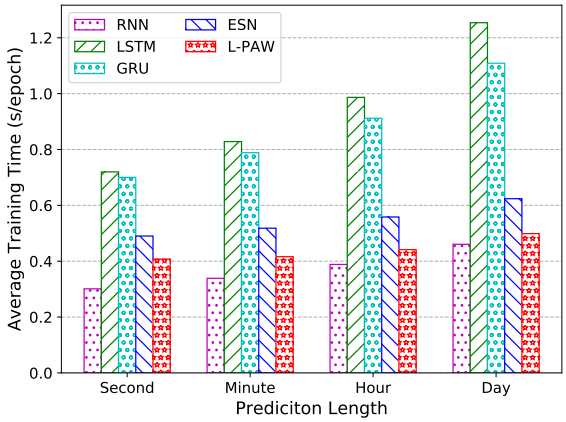


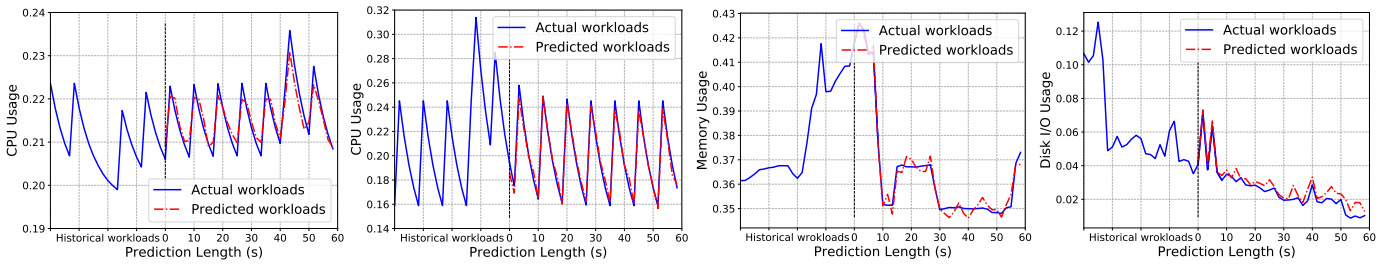
Fig. 11. Average training time (s/epoch) of different RNN-based methods with various levels of prediction length.

Next, we compare the learning efficiency of the proposed L-PAW and other RNN-based methods for workload prediction with various levels of prediction length. As shown in Fig. 11, the values of average training time of the above methods with different levels of prediction length are recorded. RNN consumes the lowest training time at different levels of prediction length due to its simple neural network structure. However, the prediction accuracy of RNN is worse than other RNN-based methods due to the workload variations in cloud data centers, as presented in Fig. 10. When it comes to gated RNN-based methods, GRU consumes less average training time compared to LSTM due to the setting of fewer gates. However, the average training time of GRU is much higher than ESN, because ESN reduces the computational complexity by using auto-encoder for workload compression. By contrast, the L-PAW achieves less average training time than ESN through integrating TSA and GRU block into RNN. Therefore, the L-PAW can achieve a better trade-off between prediction accuracy and learning efficiency than other RNN-based methods with the increase of prediction length.

TABLE 1
Prediction Accuracy (MSE) of the L-PAW with Various Levels of Prediction Length over Different Types of Workloads

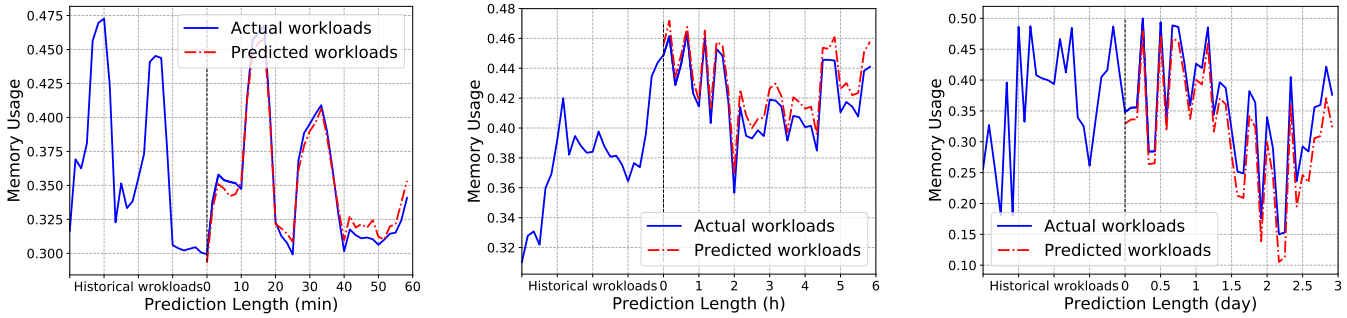
Prediction length	**MSE of workload prediction**			
	Highly auto-correlated (DUX)	***Highly periodic (DUX)***	***Highly random (Google)***	***Highly random (Alibaba)***
20 s	0.000313	0.000297	0.003887	0.003753
40 s	0.000313	0.000298	0.003889	0.003754
60 s	0.000314	0.000298	0.003893	0.003759
20 min	0.000317	0.000303	0.004061	0.003947
40 min	0.000321	0.000307	0.004232	0.004102
60 min	0.000327	0.000309	0.004306	0.004287
2 h	0.000342	0.000322	0.004628	0.004563
4 h	0.000359	0.000339	0.005169	0.004998
6 h	0.000378	0.000361	0.005746	0.005574
1 day	0.000526	0.000509	0.006812	0.006693
2 day	0.000682	0.000656	0.009123	0.008867
3 day	0.000767	0.000728	0.011598	0.011285

Next, we evaluate the performance of the proposed L-PAW for workload prediction with various levels of prediction length over different types of workloads, including the highly random, highly autocorrelated, and highly periodic workloads, respectively. As shown in Table 1, with the increase of prediction length, the L-PAW can achieve and maintain very high prediction accuracy in terms of MSE when workloads (indexed by CPU usage) are highly auto-correlated or highly periodic. While dealing with highly random workloads, the L-PAW can still obtain good prediction results although the workload variations bring great difficulty to workload prediction. For example, Fig. 12 depicts the performance of L-PAW over different types of workloads at second-level prediction, where the L-PAW can achieve a highly accurate workload prediction from the perspectives of CPU, memory, and disk I/O usage. As shown in Figs. 13 and 14, the L-PAW still exhibits excellent prediction accuracy in response to the highly random workloads from



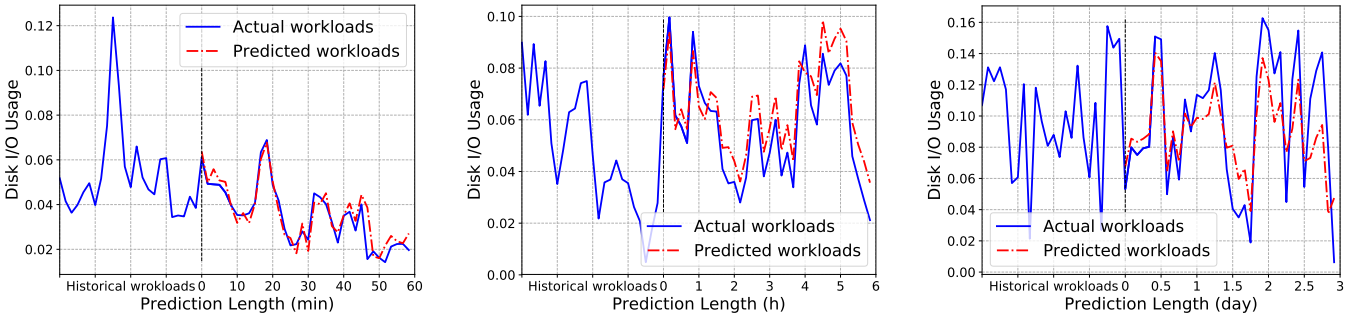
(a) Highly autocorrelated (DUX). (b) Highly periodic (DUX). (c) Highly random (Google). (d) Highly random (Alibaba).

Fig. 12. Performance display of the L-PAW over different types of workloads at second-level prediction.



(a) Minute-level prediction. (b) Hour-level prediction. (c) Day-level prediction.

Fig. 13. Performance display of the L-PAW over highly random workloads of Google cloud data centers with different levels of prediction length.



(a) Minute-level prediction. (b) Hour-level prediction. (c) Day-level prediction.

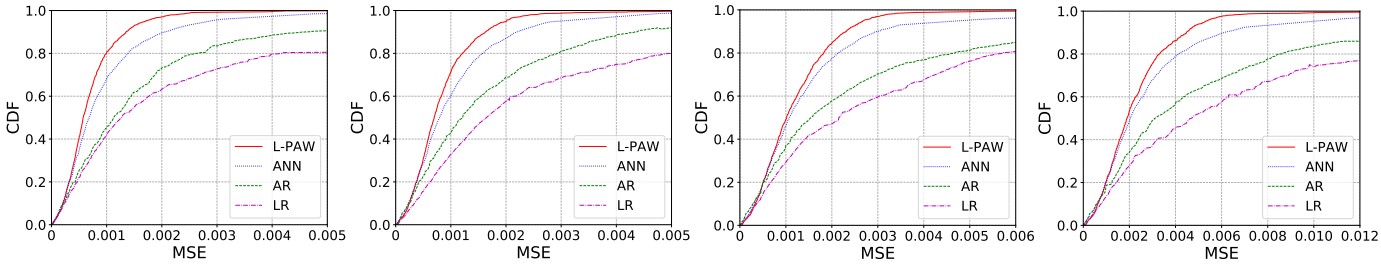
Fig. 14. Performance display of the L-PAW over highly random workloads of Alibaba cloud data centers with different levels of prediction length.

Google (memory usage) and Alibaba (disk I/O usage) cloud data centers, respectively. Even for the day-level prediction, the L-PAW can predict the future tendency of workloads accurately. Therefore, the above results demonstrate the strong adaptability of the L-PAW for handling different types of workloads with various levels of prediction length.

Finally, we compare the proposed L-PAW with other classic methods for workload prediction, including auto-regression (AR) [15], linear regression (LR) [16] and artificial neural network (ANN) [17], in terms of prediction accuracy measured by MSE. Fig. 15 illustrates the cumulative distribution function (CDF) of MSE under different methods of workload prediction with various levels of prediction length using Google datasets. As shown in Fig. 15(a), for the second-level prediction, the value of MSE achieved by the L-PAW is lower than all the other methods when CDF is close to 1. With the increase of prediction length, the L-PAW achieves more significant performance improvements compared to other classic methods, as shown in Figs. 15(b) and (d). This is because that these classic methods cannot effectively make the long-term prediction for highly random workloads without obvious regularity. By contrast, the L-PAW can better address this problem, because it can extract the representative features from raw workload data through using the TSA and capture long-term dependencies of memory from historical workloads through integrating the GRU.

7 Conclusions

The adaptive and effective workload prediction is essential to efficient resource provisioning in cloud computing. However, workload prediction is confronted with important challenges caused by the high variance and high dimensionality of cloud workloads. In this paper, we first design a TSA



(a) Second-level prediction. (b) Minute-level prediction. (c) Hour-level prediction. (d) Day-level prediction.

Fig. 15. Performance comparison between the L-PAW and other classic methods for workload prediction.

to efficiently extract the essential feature representations of workloads from original high-dimensional workload data. Next, an L-PAW was proposed through integrating TSA and GRU block, in order to achieve the adaptive and accurate prediction for highly-variable workloads. The extensive experiments using real-world workload datasets from Google and Alibaba cloud data centers and the DUX-based cluster demonstrate that the L-PAW yields high prediction accuracy measured by MSE for highly autocorrelated, highly periodic and highly random workloads. Furthermore, the prediction errors only produce small increases with the growth of prediction length, which verifies the strong adaptability of the L-PAW. Moreover, the L-PAW outperforms the classic RNN, LSTM, GRU, and ESN for workload prediction with the improvements in MSE, while achieving high learning efficiency. In addition, with the increase of prediction length, larger improvements of prediction accuracy are achieved by the L-PAW compared to other classic methods, which reveals the ability of the L-PAW for addressing the problem of long-term memory dependencies in cloud workload prediction. Based on the accurate and efficient prediction for cloud workloads, our future work is to explore an adaptive strategy for resource provisioning with deep reinforcement learning (DRL) in response to the complex and dynamic environment of cloud computing.

REFERENCES

[1] A. Alsarhan, A. Itradat, A. Y. Al-Dubai, A. Y. Zomaya, and G. Min, "Adaptive resource allocation and provisioning in multi-service cloud environments," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 29, no. 1, pp. 31–42, 2018.

[2] Z. Wang, M. M. Hayat, N. Ghani, and K. B. Shaban, "Optimizing cloud-service performance: Efficient resource provisioning via optimal workload allocation," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 28, no. 6, pp. 1689–1702, 2017.

[3] Z. Chen, J. Hu, and G. Min, "Learning-based resource allocation in cloud data center using advantage actor-critic," in *53rd International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2019.

[4] F. Xu, H. Zheng, H. Jiang, W. Shao, H. Liu, and Z. Zhou, "Cost-effective cloud server provisioning for predictable performance of big data analytics," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 30, no. 5, pp. 1036–1051, 2018.

[5] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format+ schema," *Google Inc., White Paper*, pp. 1–14, 2011.

[6] P. A. Dinda, "The statistical properties of host load," *Scientific Programming*, vol. 7, no. 3-4, pp. 211–229, 1999.

[7] J. Guo, Z. Chang, S. Wang, H. Ding, Y. Feng, L. Mao, and Y. Bao, "Who limits the resource efficiency of my datacenter: an analysis of alibaba datacenter traces," in *27th International Symposium on Quality of Service (IWQoS)*, pp. 1–10, ACM, 2019.

[8] Q. Zhang, L. T. Yang, Z. Yan, Z. Chen, and P. Li, "An efficient deep learning model to predict cloud workload for industry informatics," *IEEE Transactions on Industrial Informatics (TII)*, vol. 14, no. 7, pp. 3170–3178, 2018.

[9] Y. Hsu, K. Matsuda, and M. Matsuoka, "Self-aware workload forecasting in data center power prediction," in *18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pp. 321–330, IEEE/ACM, 2018.

[10] Z. Zhou, F. Liu, Z. Li, and H. Jin, "When smart grid meets geo-distributed cloud: An auction approach to datacenter demand response," in *34th International Conference on Computer Communications (INFOCOM)*, pp. 2650–2658, IEEE, 2015.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[12] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[13] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *32nd International Conference on Machine Learning (ICML)*, pp. 2342–2350, 2015.

[14] I. Jolliffe, "Principal component analysis," in *International Encyclopedia of Statistical Science*, pp. 1094–1096, Springer, 2011.

[15] Y. Hu, B. Deng, F. Peng, and D. Wang, "Workload prediction for cloud computing elasticity mechanism," in *International Conference on Cloud Computing and Big Data Analysis*, pp. 244–249, IEEE, 2016.

[16] X. Tang, X. Liao, J. Zheng, and X. Yang, "Energy efficient job scheduling with workload prediction on cloud data center," *Cluster Computing*, vol. 21, no. 3, pp. 1581–1593, 2018.

[17] J. Kumar and A. K. Singh, "Workload prediction in cloud using artificial neural network and adaptive differential evolution," *Future Generation Computer Systems (FGCS)*, vol. 81, pp. 41–52, 2018.

[18] Y. Yu, V. Jindal, F. Bastani, F. Li, and I. Yen, "Improving the smartness of cloud management via machine learning based workload prediction," in *42nd Annual Computer Software and Applications Conference (COMPSAC)*, pp. 38–44, IEEE, 2018.

[19] S. Salaria, A. Drozd, A. Podobas, and S. Matsuoka, "Predicting performance using collaborative filtering," in *International Conference on Cluster Computing (CLUSTER)*, pp. 504–514, IEEE, 2018.

[20] K. Mason, M. Duggan, E. Barrett, J. Duggan, and E. Howley, "Predicting host cpu utilization in the cloud using evolutionary neural networks," *Future Generation Computer Systems (FGCS)*, vol. 86, pp. 162–173, 2018.

[21] G. Kaur, A. Bala, and I. Chana, "An intelligent regressive ensemble approach for predicting resource usage in cloud computing," *Journal of Parallel and Distributed Computing (JPDC)*, vol. 123, pp. 1–12, 2019.

[22] W. Zhang, B. Li, D. Zhao, F. Gong, and Q. Lu, "Workload prediction for cloud cluster using a recurrent neural network," in *International Conference on Identification, Information and Knowledge in the Internet of Things*, pp. 104–109, IEEE, 2016.

[23] Z. Huang, J. Peng, H. Lian, J. Guo, and W. Qiu, "Deep recurrent model for server load and performance prediction in data center," *Complexity*, vol. 2017, no. 99, pp. 1–10, 2017.

[24] M. Duggan, K. Mason, J. Duggan, E. Howley, and E. Barrett, "Predicting host cpu utilization in cloud computing using recurrent neural networks," in *12th International Conference for Internet Technology and Secured Transactions*, pp. 67–72, IEEE, 2017.

[25] H. Shi, M. Xu, and R. Li, "Deep learning for household load forecastinga novel pooling deep rnn," *IEEE Transactions on Smart Grid (TSG)*, vol. 9, no. 5, pp. 5271–5280, 2018.

[26] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on lstm recurrent neural network," *IEEE Transactions on Smart Grid (TSG)*, vol. 10, no. 1, pp. 841–851, 2019.
[27] 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*, no. 2, pp. 1–15, 2017.
[28] 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.
[29] S. Kumar, N. Muthiyan, S. Gupta, A. Dileep, and A. Nigam, "Association learning based hybrid model for cloud workload prediction," in *31st International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2018.
[30] Y. Guo and W. Yao, "Applying gated recurrent units approaches for workload prediction," in *16th Network Operations and Management Symposium (NOMS)*, pp. 1–6, IEEE, 2018.
[31] H. Liu, "A measurement study of server utilization in public clouds," in *9th International Conference on Dependable, Autonomic and Secure Computing (DASC)*, pp. 435–442, IEEE, 2011.
[32] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini, "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms," in *26th Symposium on Operating Systems Principles (SOSP)*, pp. 153–167, ACM, 2017.
[33] M. Ranjbari and J. A. Torkestani, "A learning automata-based algorithm for energy and sla efficient consolidation of virtual machines in cloud data centers," *Journal of Parallel and Distributed Computing (JPDC)*, vol. 113, pp. 55–62, 2018.
[34] C. Delimitrou and C. Kozyrakis, "Paragon: Qos-aware scheduling for heterogeneous datacenters," in *18th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, vol. 48, pp. 77–88, ACM, 2013.
[35] K. Wang, C. Xu, Y. Zhang, S. Guo, and A. Zomaya, "Robust big data analytics for electricity price forecasting in the smart grid," *IEEE Transactions on Big Data (TBD)*, vol. 5, no. 1, pp. 34–45, 2017.
[36] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, *et al.*, "Tensorflow: a system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 265–283, USENIX, 2016.

Zheyi Chen is currently a Ph.D. candidate in Computer Science at the University of Exeter. He received his M.Sc. degree in Computer Science from Tsinghua University, China, in 2017, and B.Sc. degree in Computer Science from Shanxi University, China, in 2014. His research interests include cloud computing, mobile edge computing, deep learning, and resource optimization.

Jia Hu is a Lecturer in Computer Science at the University of Exeter. He received his Ph.D. degree in Computer Science from the University of Bradford, UK, in 2010, and M.Eng. and B.Eng. degrees in Electronic Engineering from Huazhong University of Science and Technology, China, in 2006 and 2004, respectively. His research interests include mobile and ubiquitous computing, resource optimization, applied machine learning, and network security.

Geyong Min is a Professor of High Performance Computing and Networking in the Department of Computer Science within the College of Engineering, Mathematics and Physical Sciences at the University of Exeter, United Kingdom. He received the PhD degree in Computing Science from the University of Glasgow, United Kingdom, in 2003, and the B.Sc. degree in Computer Science from Huazhong University of Science and Technology, China, in 1995. His research interests include future Internet, computer networks, wireless communications, multimedia systems, information security, high-performance computing, ubiquitous computing, modelling and performance engineering.

Albert Y. Zomaya is currently the Chair Professor of High Performance Computing & Networking in the School of Computer Science, University of Sydney. He is also the Director of the Centre for Distributed and High Performance Computing. He published more than 600 scientific papers and articles and is author, co-author or editor of more than 20 books.

He is the Founding Editor in Chief of the IEEE Transactions on Sustainable Computing and previously he served as Editor in Chief for the IEEE Transactions on Computers (2011-2014). Currently, Professor Zomaya serves as an associate editor for 22 leading journals, such as, the ACM Computing Surveys, IEEE Transactions on Cloud Computing, and ACM Transactions on Internet Technology. He delivered more than 190 keynote addresses, invited seminars, and media briefings and has been actively involved, in a variety of capacities, in the organization of more than 700 conferences.

Professor Zomaya is the recipient of the IEEE Technical Committee on Parallel Processing Outstanding Service Award (2011), the IEEE Technical Committee on Scalable Computing Medal for Excellence in Scalable Computing (2011), the IEEE Computer Society Technical Achievement Award (2014), the ACM MSWIM Reginald A. Fessenden Award (2017), and the New South Wales Premiers Prize of Excellence in Engineering and Information and Communications Technology (2019). He is a Chartered Engineer, a Fellow of AAAS, IEEE, IET (UK), an Elected Member of Academia Europaea, and an IEEE Computer Societys Golden Core member. Professor Zomayas research interests lie in parallel and distributed computing, networking, and complex systems.

Tarek El-Ghazawi is a Professor in the Department of Electrical and Computer Engineering at The George Washington University, where he leads the university-wide Strategic Academic Program in High-Performance Computing. He is the founding director of The GW Institute for Massively Parallel Applications and Computing Technologies (IMPACT) and a founding Co-Director of the NSF Industry/University Center for High-Performance Reconfigurable Computing (CHREC). He received his Ph.D. degree in Electrical and Computer Engineering from New Mexico State University in 1988. His research interests include high-performance computing, parallel computer architectures, high-performance I/O, reconfigurable computing, experimental performance evaluations, computer vision, and remote sensing. Prof. El-Ghazawi has published over 200 refereed research papers and book chapters in these areas and his research has been supported by DoD/DARPA, NASA, NSF, and also industry, including IBM and SGI. He is a fellow of the IEEE.