esDNN: Deep Neural Network based Multivariate Workload Prediction Approach in Cloud Computing Environments

MINXIAN XU and CHENGHAO SONG, Shenzhen Institute of Advanced Technology, CAS, China
HUAMING WU, Tianjin University, China
SUKHPAL SINGH GILL, Queen Mary University of London, UK
KEJIANG YE*, Shenzhen Institute of Advanced Technology, CAS, China
CHENGZHONG XU, State Key Lab of IOTSC, University of Macau, China

Cloud computing has been regarded as a successful paradigm for IT industry by providing benefits for both service providers and customers. In spite of the advantages, cloud computing also suffers from distinct challenges, and one of them is the inefficient resource provisioning for dynamic workloads. Accurate workload predictions for cloud computing can support efficient resource provisioning and avoid resource wastage. However, due to the high-dimensional and high-variable features of cloud workloads, it is difficult to predict the workloads effectively and accurately. The current dominant work for cloud workload prediction is based on regression approaches or recurrent neural networks, which fail to capture the long-term variance of workloads. To address the challenges and overcome the limitations of existing works, we proposed an ***e***fficient ***s***upervised learning-based ***D***eep ***N***eural ***N***etwork (***esDNN***) approach for cloud workload prediction. Firstly, we utilize a sliding window to convert the multivariate data into supervised learning time series that allow deep learning for processing. Then we apply a revised Gated Recurrent Unit (GRU) to achieve accurate prediction. To show the effectiveness of esDNN, we also conduct comprehensive experiments based on realistic traces derived from Alibaba and Google cloud data centers. The experimental results demonstrate that esDNN can accurately and efficiently predict cloud workloads. Compared with the state-of-the-art baselines, esDNN can reduce the mean square errors significantly, e.g. 15% than the approach using GRU only. We also apply esDNN for machines auto-scaling, which illustrates that esDNN can reduce the number of active hosts efficiently, thus the costs of service providers can be optimized.

CCS Concepts: • **General Literature**; • **Distributed Parallel and Cluster Computing**; • **System and Software**; • **Management of Cloud Computing Systems**;

Additional Key Words and Phrases: cloud Computing, workloads prediction, supervised learning, gate recurrent unit, auto-scaling

ACM Reference Format:
Minxian Xu, Chenghao Song, Huaming Wu, Sukhpal Singh Gill, Kejiang Ye, and Chengzhong Xu. 2022. esDNN: Deep Neural Network based Multivariate Workload Prediction Approach in Cloud Computing Environments. *ACM Trans. Internet Technol.* 1, 1, Article 1 (March 2022), 24 pages.

1 INTRODUCTION

Today's organizations and enterprises are becoming more dependent upon information technologies with cloud services that are deployed in cloud data centers [1, 2]. Cloud services offer significant

*Corresponding author: Kejiang Ye

Authors' addresses: Minxian Xu, mx.xu@siat.ac.cn; Chenghao Song, ch.song@siat.ac.cn, Shenzhen Institute of Advanced Technology, CAS, Shenzhen, Guangdong, China, 518000; Huaming Wu, Tianjin University, Tianjin, China, whming@tju.edu.cn; Sukhpal Singh Gill, Queen Mary University of London, London, UK; Kejiang Ye, Shenzhen Institute of Advanced Technology, CAS, Shenzhen, Guangdong, China; Chengzhong Xu, State Key Lab of IOTSC, University of Macau, Macau, China.

1533-5399/2022/3-ART1 $0

benefits for both customers and service providers [3]. The customers can access the services with high availability, and the service providers can take advantage of elasticity and low management costs of infrastructure. The pay-as-you-go pricing model is also a dominant benefit that promotes the fast development of cloud computing [4]. Due to these benefits, large cloud service providers, e.g. Amazon, Google and Microsoft have established large-scale data centers to provide resources for their services and a great number of companies have started to migrate their local services to the cloud [5].

Although cloud computing is featured with these attractive benefits, some unpredictable situations, e.g. workload bursts can lead to resources being insufficient. The unmatched resources for workloads can also waste resources or degrade performance, for instance, more resources are provisioned than required when workloads are at a low level and only limited resources are offered when workloads are increasing dramatically [6]. Therefore, to improve the resource usage, predicting workloads in an accurate manner is required. With the effective prediction of future workloads, the service provider can plan resources in a more efficient and rational way by allocating or de-allocating resources in advance [7].

However, it is not an easy job to predict cloud workloads efficiently and accurately due to their native characteristics. Cloud workloads have high variance and high dimensionality, which make them difficult to forecast. High variance represents that the number of workloads and their demanded resources can change dramatically. According to the analysis of Alibaba cloud data centers, the average resource utilization can range from 5% to 80% [8]. And in Google cloud data centers, workloads can change randomly during a specific observation period. As for high dimensionality, it represents that cloud workload traces record a great amount of information and different specification of machines, which needs to extract the necessary and valuable information for the training model.

To address the high variance challenge of cloud workloads, the pattern of workloads, as well as the relationship with time series, should be learned and exploited to design efficient and accurate prediction algorithms to fit with the variances of workloads. As for the high dimensionality challenge, the dataset can be further analyzed to extract the necessary data while assuring the prediction accuracy.

A significant amount of research has been devoted to cloud workload prediction. Traditional approaches are mostly based on the regression methods, heuristic algorithms and traditional neural network approaches. Traditional neural networks generally refer to shallow networks that contain only several layers, such as Multi-layer Perceptron (MLP) and Radial Basis Function (RBF). However, these approaches can only work effectively for the workloads with obvious patterns, e.g. for small-scale data centers for ordinary companies or organizations. For the large-scale public cloud data centers, these approaches can not obtain high prediction accuracy. The main reason is that the regression methods and simple neural networks cannot capture the complicated correlation of workloads. Therefore, to achieve higher accuracy, more complicated neural networks can be applied to take full advantage of the correlations of neurons.

As a representative of neural network-based approaches, the Recurrent Neural Network (RNN) [9] has been applied to predict cloud workloads as it has the feature to model the changes with time series. RNN can use its memory to process a set of inputs in sequence. However, it is inefficient for RNN to learn long-term memory dependencies because of the gradient vanishing. To overcome this limitation, some revised RNN, including Long Short-Term Memory (LSTM) [10] and Gated Recurrent Unit (GRU) [11] have been proposed, which have demonstrated a strong capacity to learn long-term memory dependencies. Compared with LSTM, GRU has demonstrated better prediction accuracy and learning efficiency in practice. Thus, in this work, we apply a GRU-based approach to capture the variance of cloud workloads.

1.1 Motivation and Our Contributions

To address the high dimensionality challenge, extraction of features of the original data is required. Our main motivations are as follows:

- Some approaches including Principal Component Analysis (PCA) [12] and auto-encoder [13] have been investigated, which can reduce dimension largely, while the accuracy is degraded as some features have been ignored.
- Traditional machine learning models can only show the mapping relationship between the source data and the target data, however, the time relationship cannot be extracted and exploited.
- When predicting long periods, the dominant time-series data prediction approaches based on LSTM and RNN have the limitations of gradient disappearance and explosion.

To address the aforementioned challenges for cloud workload prediction, we first extract some key features from the realistic traces derived from the cloud data center, and then convert the multivariate time series into supervised learning time series [14] for further training with our designed training algorithm based on GRU. Our objective is to achieve efficient and accurate predictions for highly variable and high dimensional cloud workloads to finally optimize the resource usage in cloud computing environments.

The main **contributions** of this paper are summarized as follows:

- The sliding window for Multivariate Time series Forecasting (S-MTF) is designed to convert multivariate time series into supervised learning time series for multivariate workloads and keep sufficient information. The S-MTF can reorganize the time series to sample X and label Y and model the correlation between predicted data, which can use algorithms based on Deep Neural Network (DNN) to achieve predictions.
- An ***e***fficient ***s***upervised learning-based ***D***eep ***N***eural ***N***etwork (***esDNN***) algorithm is proposed for cloud workload prediction to learn and capture the features of historical data and accurately predict future workloads. The proposed algorithm can adapt to the variances of workloads by updating the gates of GRU and overcome the limitations of gradient disappearance and explosion.
- Comprehensive experiments are conducted by using realistic data derived from Alibaba and Google cloud data centers to evaluate the performance of esDNN. The results demonstrate that the proposed approach can achieve better prediction accuracy than state-of-the-art algorithms. Experiments also show that the proposed approach can be applied for auto-scaling scenarios to improve resource provisioning.

1.2 Article Organization

The rest of the paper is organized as follows: Section 2 discusses the related work for workload prediction in cloud computing environments. Section 3 depicts the system model of our proposed approach, followed by system problem statement. The proposed algorithm based on DNN is introduced in Section 4. Section 5 introduces the details of our experiments that apply dataset derived from realistic traces to predict workloads, and demonstrate the feasibility of our approach to improve the resource provisioning of cloud data centers. Finally, conclusions along with the future directions are given in Section 6.

2 RELATED WORK

Many researchers have conducted research on workload prediction. The main contributions for cloud workload prediction can be classified as regression-based and learning-based approaches.

The regression-based approaches mainly include linear regression, auto-regression and other traditional regression-based approaches. While for the learning-based approaches, both the traditional approaches based on machine learning and some updated methodologies based on deep learning have been investigated.

2.1 Regression-based Approaches for Cloud Workload Prediction

Calheiros et al. [15] proposed an approach based on auto-regression to predict future workloads by using requests of web applications. The proposed approach can achieve high accuracy in resource utilization and QoS prediction. Yang et al. [16] introduced an approach based on linear regression for workload prediction to satisfy Service Level Agreement (SLA) and reduce scaling costs. Based on the prediction data, the auto-scaling mechanism can be further applied to optimize virtualized resource usage. Centinski et al. [17] combined statistical and machine learning methods together to improve workload prediction for cloud applications. The training method is utilized to learn the dominant system parameters of the influence application, and the prediction method is based on the regression approach. Singh et al. [18] presented a combined algorithm based on linear regression and support vector machine for workload prediction of web applications. A workload classifier was also proposed to select the model based on workloads features. Liu et al. [19] introduced an adaptive workload prediction approach based on workloads classification, in which different prediction models can be assigned to the different categorized workloads. Bi et al. [20] proposed a prediction method that integrates Savitzky-Golay filter and wavelet decomposition with stochastic configuration networks to predict workloads.

These regression-based approaches have proven their effectiveness in workload prediction. However, most of these approaches are only suitable for workloads with obvious patterns, e.g. Wikipedia workloads with fixed daily tendencies. The modern cloud workloads with high variance make these approaches hard to represent correlations between different parameters. Besides, these approaches were applied to high-performance computing workloads, small-scale data centers or synthetic workloads, which have lower variance compared with cloud workloads. Therefore, to efficiently capture the characteristics of cloud workloads, more advanced learning approaches, e.g. machine learning and deep learning-based methodologies have been investigated.

2.2 Learning-based Approaches for Cloud Workload Prediction

Kumar et al. [21] applied a neural network and self-adaptive differential evolution algorithm to learn and extract the pattern from workloads. This evolution-based approach can reduce the prediction error by searching a large solution space, thereby minimizing the effects of initial solution choice. Zheng et al. [22] presented a deep learning model based on canonical polyadic decomposition to predict the usage of virtual machines for cloud workloads for industry informatics. Compared with machine learning-based approaches, deep learning-based approaches can achieve higher accuracy. Kumar et al. [23] proposed a prediction model based on LSTM and showed good performance in reducing mean square errors. Qiu et al. [24] introduced a deep learning approach to predict Virtual Machine (VM) workloads by extracting high-level features of VMs workloads and then predicting future VM workloads. Zhu et al. [25] presented an approach based on LSTM encoder-decoders network with an attention mechanism. The features of historical data are extracted via the encoder network and the attention mechanism is integrated into the decoder network. Amiri et al. [26] introduced an online learning approach to adapt resources according to workloads variations based on sequential pattern mining, which can learn new behavioral patterns rapidly. Chen et al. [7] proposed a deep learning-based approach, which includes a top-sparse auto-encoder to extract essential features of workloads and GRU to obtain an accurate and adaptive prediction for cloud workloads. Several different types of workloads have been investigated to validate the effectiveness

of the proposed approach. Eli et al. [27] presented a resource central system to collect Azure VM parameters to learn the VM behavior offline with Microsoft learning libraries and then make online resource usage prediction, which predicts the oversubscription of VM types while ensuring VM performance.

Bi et al. [28] applied bi-directional LSTM (Bi-LSTM) to predict large-scale workloads and resource consumption in the cloud computing environment. The performance of the approach has been validated with Google traces and shown better results than baselines. Karim et al. [29] proposed a hybrid approach combing RNN and Bi-LSTM to forecast CPU workload of VMs, which can improve the performance of using a single technique separately. Chen et al. [30] introduced the LSTM-based approach to predict the useful life of components to indicate system health. A support vector regression is also combined to enhance the prediction robustness and marginal utility. Results based on NASA have validated the effectiveness of the proposed approach. Singh et al. [31] proposed an evolutionary quantum neural network-based approach for cloud workloads prediction, which leverages the computational efficiency of quantum computing to encode workloads, and utilizes the neural network to estimate resource demands. The experiments with traces from cloud data centers and traditional data centers have validated the effectiveness of the proposed approach. Kim et al. [32] introduced a cloud prediction framework named CloudInsight that combines multiple predictors based on traditional machine learning techniques to enable accurate predictions for real cloud workloads. The ensemble supports dynamic and periodical optimization to handle the variations of workloads. The framework can also reduce the periods of under-provisioning and over-provisioning, thus improving system efficiency.

The deep learning based approaches have been applied in predictions in many areas, such as communication, economic market, and pedestrian motion. Sun et al. [33] proposed LSTM-based approach to predict link quality confidence interval for wireless communication under a smart grid environment. A wavelet denoising algorithm has been applied to decompose the signal-to-noise ratio time series into the deterministic and stochastic ones to train two LSTM neural networks. Li et al. [34] introduced a recurrent attention and interaction model to predict pedestrian trajectories, which includes several modules to achieve precise prediction collaboratively. The introduced approach can comprehensively mine the spatio-temporal information to model attention mechanisms, interactions, and multimodality of pedestrian motion. Barra et al. [35] presented an approach to forecast market behavior by encoding time series to Gramian angular fields images based on neural networks. Qiao et al. [36] proposed an approach based on a neural network to model the uncertain nonlinear systems by utilizing a distance concentration algorithm to increase prediction accuracy and reduce computation time. However, these approaches are not focusing on cloud workloads prediction.

To summarize, most of the learning-based approaches are based on machine learning algorithms or traditional RNN, which either cannot exploit the long-term memory dependencies or address the gradient vanishing challenge. Thus, it is also difficult for them to predict cloud workloads accurately. Only limited research has paid attention to GRU, which is an improved version of RNN and can address the gradient vanishing challenge to achieve better accuracy. For instance, Chen et al. [7] applied GRU for cloud workload prediction, however, they also apply the auto-encoder approach to compress the dimensionality of the original data. Although the auto-encoder approach can address the high dimensionality, the accuracy is also undermined since the full data is not utilized to capture the whole features of workloads.

2.3 Critical Analysis

The current paper contributes to the growing body of work in the cloud workload prediction area. The comparison of our proposed approach and the related work is summarized in Table 1. To solve

Table 1. Comparison of related work

Approach	Technique							Data Preprocessing		Predicted Resources			Workloads			Performance Metrics			
	Regression	Machine Learning	Episode Mining	Deep Learning				Auto-encoder	Sliding window	VM utilization	Server Utilization	QoS	Realistic		Synthetic	MSE	RMSE	MAPE	CDF
				DNN	DBN	LSTM	GRU						Cloud Data Centers	Traditional Data Centers					
Calheiros et al. [15]	√										√	√		√				√	
Amiri et al. [26]			√							√		√		√	√			√	
Ceninski et al. [17]		√										√		√		√			
Kumar et al. [21]				√						√		√		√		√			
Kumar et al. [23]						√					√			√		√			
Qiu et al. [24]					√					√	√			√	√			√	
Rodrigo et al. [15]	√									√		√		√				√	√
Singh et al. [18]	√	√							√	√		√		√		√	√	√	
Yang et al. [16]	√									√	√			√	√				√
Zhang et al. [22]		√						√		√				√			√	√	
Zhu et al. [25]						√		√			√		√			√	√	√	
Bi et al. [20]				√							√		√			√			
Liu et al. [19]	√										√		√			√			
Eli et al. [27]		√								√	√		√						√
Sun et al. [33]						√						√			√	√			
Li et al. [34]						√													
Barra et al. [35]				√															
Qiao et al. [36]				√													√		
Bi et al. [28]	√					√					√	√	√			√			
Singh et al. [31]				√							√		√	√			√		
Kim et al. [32]		√									√	√	√	√			√	√	√
Karim et al. [29]						√	√		√	√				√		√	√	√	
Chen et al. [30]	√					√								√			√		
esDNN (This Work)							√		√		√		√			√	√	√	√

the aforementioned challenges, e.g. high-dimensional problems and multivariate problems, we apply GRU to capture the long-term memory dependencies to address the high variance of cloud workloads, thereby achieving high accuracy prediction of cloud workloads. We also apply a sliding window for multivariate time series prediction to convert the original time series into supervised learning time series to address the high dimensionality and further achieve higher accuracy. From the technique perspective, our GRU-based approach is advanced in prediction compared with traditional regression and machine learning based approaches, and aims to overcome the limitation of gradient disappearance and explosion that exist in approaches like LSTM. From the data preprocessing perspective, our approach focusing on a sliding window to take advantage of full data information and the correlation between the predicted data rather than only extracting part of data like in auto-encoded based approach. We also validated our approach based on realistic traces of Google and Alibaba and multiple metrics have been evaluated comprehensively.

3 SYSTEM MODEL

In this section, we introduce our system model and optimization objective. In our system model, we aim to offer an efficient and accurate prediction model that the service providers can apply to predict future workloads. Thus, the resource usage can be optimized to reduce their costs, e.g. integrating the model with auto-scaling to reduce the number of active hosts.

It is not easy to predict cloud workloads as they can change dramatically within a short time and the pattern is also difficult to capture precisely. For example, workloads in every 5 minutes from the dataset of Alibaba can vary significantly [8]. The cloud workloads are tightly coupled with time series, and it is inefficient to get accurate prediction results from a simple regression model or univariate time predictions. Multivariate time series can contain more dynamic information than univariate time series. For instance, the data in multivariate time series forecasting can have certain correlations, such as CPU usage and memory usage in workload forecasting. Therefore, we built a multivariate time series forecasting model to predict highly random workloads and use the real-world dataset to verify the accuracy of the model. In our prediction model, we use CPU usage as our standard for measuring the prediction results. Fig. 1 shows the main components and flow of the system model.

Step 1: Data Preprocessing. This step is equipped with workloads preprocessing component and data cleaning component, which processes the raw data derived from the realistic cloud traces. With the raw data of cloud workloads, we first remove the columns that contain empty data.

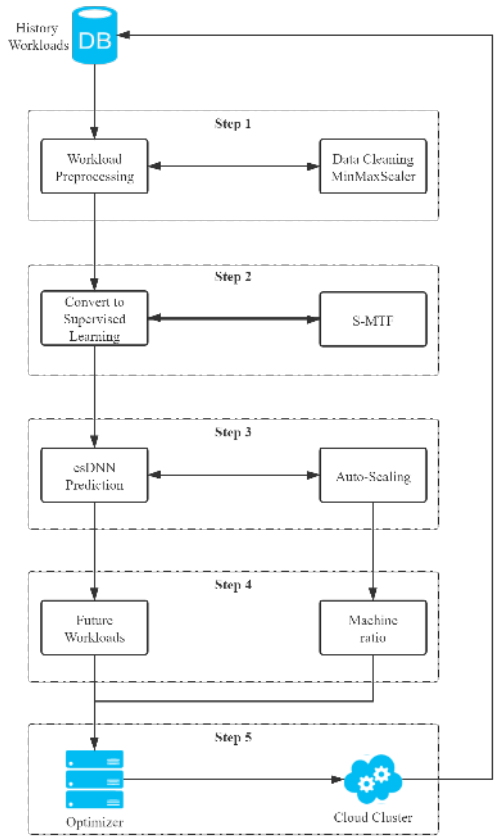


Fig. 1. Multivariate time series prediction model for cloud workloads

Because whether it is to use the zero-filling scheme or simply ignore these data, they will have a negative impact on our forecast data. Afterwards, we classify the dataset by time, then calculate the average value of each parameter with the same timestamp. Next, we normalized the Alibaba dataset and Google dataset. Normalization is a dimensionless processing method that makes the absolute value of the physical system value into a certain relative value relationship. From the perspective of model optimization, normalization can not only improve the convergence speed of the model but also improve the accuracy of prediction. The normalization method has two forms, one is to change the number to a decimal between (0, 1), and the other is to change the dimensional expression to a non-dimensional expression and become a scalar. In this article, the former is chosen as the normalization method, and we use MinMaxScaler to achieve this function. The MinMaxScaler operation is based on the min-max scaling method as follows:

$$X_{\text{std}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} - X_{\text{min}}, \tag{1}$$

$$X_{\text{scaled}} = X_{\text{std}} * (max - min) + min. \tag{2}$$

We apply the MinMaxScaler to transform features with default configurations and scale each feature to be a value between min and max. The X represents the set of the data to be processed, and the

X_{min} and X_{max} are the minimum and maximum data in the dataset, and the final processed data is represented by X_{scaled}. To be noted, in this work, the predicted value is resource utilization, which ranges from 0.0 to 1.0, and the MinMaxScaler can handle these data well. As for the missing data, they will be filled with the data in the previous time slot.

Step 2: Supervised Learning Conversion. The difference between supervised learning and unsupervised learning lies in whether there are labels of samples for training. In supervised learning, it has labeled training samples. It trains through the existing training samples to obtain an optimized model and then uses this model to map all inputs to the corresponding outputs, thereby realizing data prediction and classification. For unsupervised learning, there are no pre-labeled training samples. In our system model, we use the supervised learning transfer function to convert the multivariate time series prediction problem into a supervised learning problem based on [14]. More details about the transfer function will be introduced in Section 4. The key motivation is to use the normalized dataset as the input of the transfer function, and reframe the time series datasets as supervised learning datasets. To achieve this, we split the dataset into a training set and a validation set. After that, the dataset is divided into sample X and its corresponding label Y. With these conversion operations, we can transform the time series forecasting problem into a supervised learning-based time series problem.

Step 3: Model Construction. In this step, our system model focuses on the construction of deep learning networks and establishes an optimization model for cloud workload prediction based on the preprocessed data. The preprocessed data are considered as input and the output is the optimized parameters of the model as well as the evaluation metrics, e.g. mean square errors. In this step, the hyperparameters of the deep learning network should also be defined, e.g. the number of layers, number of neurons, and types of network. Our proposed network model is derived from GRU and more design details will be given in the following sections. By predicting the cloud workloads, we aim to obtain future resource usage and thus the optimization of the number of active machines can be optimized by auto-scaling approaches, which will be coordinately achieved with the next step.

Step 4 and Step 5: Model Deployment and System Adaption. These two steps focus on utilizing the models for workload prediction or other system optimization purposes. In the actual workload prediction, there is a time interval between two consecutive predictions, which means that the sequence prediction is based on a discrete time series dataset. For the Alibaba dataset, the prediction interval is usually 10 seconds, while the time prediction interval of Google is 5 minutes. We apply this time interval as the prediction unit. Based on the trained model in Step 3, in this step, the system model can obtain the predicted future workloads and adjust the number of machines by applying auto-scaling. With the predicted data, the realistic system can dynamically adapt the resource provisioning for the system, e.g. adding or removing machines physically, which requires the use of Application Programming Interfaces (APIs) provided by hardware.

4 ESDNN: EFFICIENT SUPERVISE LEARNING-BASED DEEP NEURAL NETWORK

This section presents our proposed approach, which is a deep learning-based approach for cloud workload prediction. To achieve efficient and accurate prediction results, a sliding window-based approach for multivariate time series prediction is applied to convert the original dataset into supervised learning-based time series data. Thereafter, a GRU-based deep learning network, named esDNN is proposed for future workload prediction.

4.1 Sliding Window for Multivariate Time Series Forecasting

Time series forecasting requires the dataset to contain a set of time-dependent data, regardless of whether the time units of the dataset are seconds, minutes, or hours. This data needs to have

Table 2. One-step univariate forecasting: raw dataset

Time	CPU utilization percentage
0	16.127
10	21.5878
20	17.3193
30	16.8287
40	18.6518

Table 3. One-step univariate forecasting: supervised learning sequence

X	Y
None	16.127
16.127	21.5878
21.5878	17.3193
17.3193	16.8287
16.8287	18.6518
18.6518	None

a minimum time unit, but it does not need to have the same time interval between two adjacent timestamps. Having clarified this concept, we can say that a time series is a sequence of numbers sorted by time index. However, only have one time series is not sufficient. In Section 3, we have introduced the definition of supervised learning. Complete supervised learning requires a sample group (X) and a label group (Y). There are two major differences compared with the work based sliding window [37] including 1) we consider the multivariate workloads to construct time series data rather than single variate; 2) we utilize the relationship between the predicted data by merging the predicted data and source data into supervised time series data together. To illustrate the conversion process more vividly, we use a small piece of sample data in the Alibaba cloud workloads dataset to show the conversion process and results. To simplify the example, we use one-step univariate forecasting.

Table 2 shows the first five rows of data from the Alibaba dataset, after we convert these time series data into supervised learning data, it will be presented in the form of Table 3, where each row data is moved up with data in the group (Y) and the time label has been removed.

For the multivariate time series datasets, we can also convert them into supervised learning datasets with sliding window approach. Similarly, we also take a small fragment from the Alibaba dataset. The difference is that in addition to Time and CPU utilization percent, we also take memory utilization percentage to reflect that this is a multivariate dataset. Here, we choose the memory utilization percentage as Y, which is considered as the label. In our model, we choose to use the one-step multivariate forecasting. Table 4 and Table 5 show the original data and the converted data, respectively.

Fig. 3 shows the typical conversion process by operating the original data. Assuming that we have the time sequence as $R(t-1), R(t)$ and $R(t+1)$, where $R(t-1)$ is the last one, $R(t)$ is the current one and $R(t+1)$ is the next one. We set elements $E(i-1), E(i)$ and $P(i+1)$ as the elements to be combined as the supervised time sequence $S(n)$. The $E(i-1)$ is from the data of $R(t-1)$, $E(i)$ is assigned by $R(t)$, and $P(i+1)$ is assigned by $R(t+1)$. The $E(i-1)$ and $E(i)$ will be the sample data and $P(i+1)$ will be the label. The other supervised time sequence, e.g. $S(n-1)$ and $S(n+1)$ can be obtained in the same way.

Table 4. One-step multivariate forecasting: raw dataset

Time	CPU util. percentage	Memory util. percentage
0	16.127	87.139
10	21.5878	87.0543
20	17.3193	86.9491
30	16.8287	86.9454
40	18.6518	86.9495
50	20.0232	86.9985
60	17.8671	86.9249

Table 5. One-step multivariate forecasting: supervised learning sequence

X1	X2	X3	Y
None	None	16.127	87.139
16.127	87.139	21.5878	87.0543
21.5878	87.0543	17.3193	86.9491
17.3193	86.9491	16.8287	86.9454
16.8287	86.9454	18.6518	86.9454
18.6518	86.9495	20.0232	86.9985
20.0232	86.9985	17.8671	86.9249
17.8671	86.9249	None	None

By this step, we have completed the application expression of multivariate time series forecasting, and what we have to do now is to abstract it into an algorithm. First, we define this algorithm as Sliding window for Multivariate Time series Forecasting (S-MTF) algorithm, which transforms a multivariate time series forecast into a supervised learning time series. The S-MTF algorithm can be applied to any time-related dataset, and it is still linearly related to time because it contains all the data of the previous moment at any time. At this point, the S-MTF is somewhat similar to LSTM, but the difference is that the forget gate of LSTM will weaken the influence from the previous moment, while S-MTF retains all the values of the previous moment, and it can be determined if you need to keep it or not. Besides, S-MTF contains the future label while using the multi-step forecasting. Furthermore, S-MTF satisfies the definition of supervised learning as it transforms time-related datasets into the sample and labeled datasets. With a more general form, Fig. 2 depicts the transformation of the S-MTF algorithm for time series and presents the supervised learning sequences obtained from the transformation in a tabular form. Algorithm 1 shows the pseudocode of the S-MTF algorithm. Before the original data are processed as the input of the algorithm, we have deleted NONE values in the dataset for the convenience of data processing, as they can influence the accuracy of the proposed algorithm.

Algorithm complexity analysis: Given that there is a set of time series data with size N, the algorithm processes the data from 1 to $N-1$ to construct the matrix S_n. To obtain all the data in S_n with 3 sub-data in each time interval, the complexity will be $O(3 \times (N-1))$, which equals $O(N)$.

4.2 esDNN Algorithm

In our deep learning networks, the input data in the training phase include the resource utilization and the corresponding time series data, e.g. at time 08:00:00 am, the CPU utilization is 20%. In the prediction phase, the input data are the resource utilization in the recent time intervals, e.g.

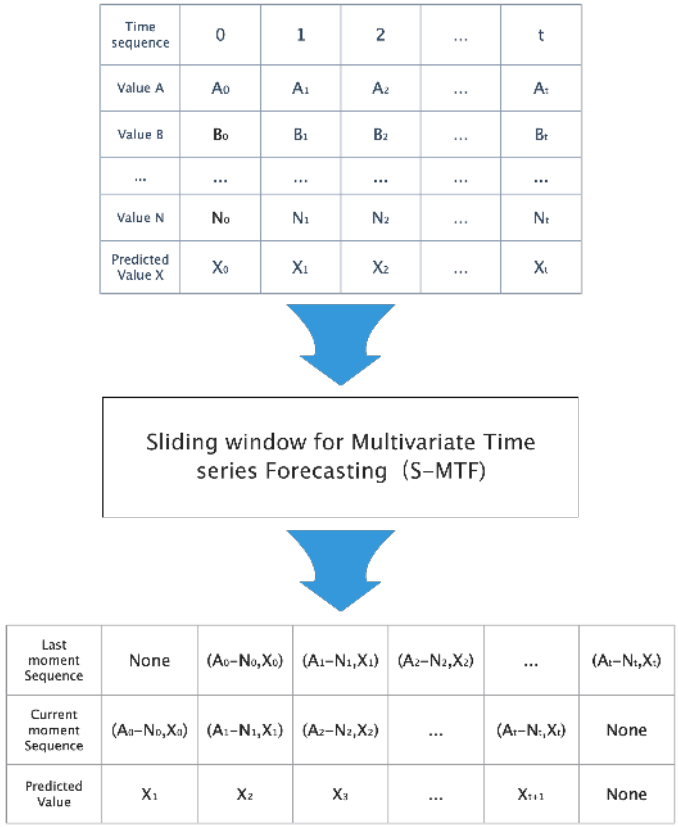


Fig. 2. The key procedures of the S-MTF algorithm

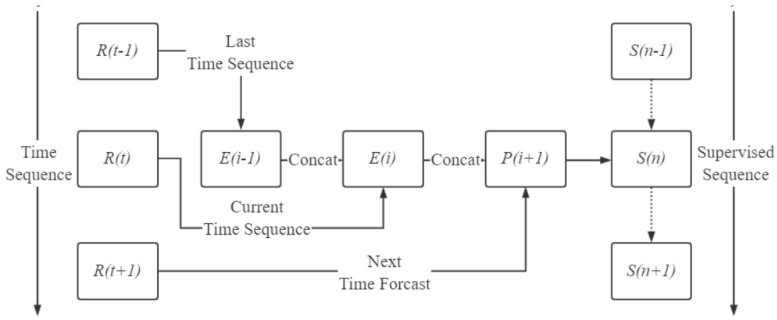


Fig. 3. Data conversion in S-MTF algorithm

the previous 5 minutes (can be configured via parameters in network model). To construct our network model, we include one layer of Convolutional Neural Network (CNN). The CNN model is usually built on the feedforward neural network model. It generally consists of Input Layer,

Algorithm 1: Sliding window for Multivariate Time series Forecasting (S-MTF)

Input : Multivariate time series dataset R_n, R_n contains a time series $R(1), R(2), \ldots, R(t), R(t+1)$, k time-related variables, and a dataset P_n to be predicted

Output : Supervised learning dataset S_n, each row of it has $2k+3$ data

Initialize an empty matrix S_n to record supervised time series data **for** t *from* 1 *to* $n-1$ **do**
    $E(t) \leftarrow R(t)$
    if $t = 1$ **then**
        Record NONE
    else
        $E(t-1) \leftarrow R(t-1)$
    end
    if $t = n-1$ **then**
        Record NONE
    else
        $P(t+1) \leftarrow R(t+1)$
    end
    Put $E(t-1)$, $E(t)$ and $P(t+1)$ together into a tuple
    Set $E(t-1)$, $E(t)$ as data in supervised time series
    Set $P(t+1)$ as label in supervised time series
    $S(t) \leftarrow \{E(t-1), E(t), P(t+1)\}$
    if $S(t)$ *contains NONE* **then**
        Delete $S(t)$
    Add $S(t)$ into S_n
end

Convolutional Layer, Pooling Layer, Non-linearity Layer and Fully Connected Layer [38]. Two-dimensional convolutional neural networks (2D CNN) are widely used in image recognition, and one-dimensional convolutional neural networks (1D CNN) are generally used in Natural Language Processing (NLP). Additionally, the one-dimensional convolutional neural networks also have the capability in processing continuous sequences. For example, when obtaining a certain feature from a shorter segment in the whole dataset, while the feature is not highly correlated with the position of the data segment in the overall dataset, in this situation, the 1D CNN can play an important role. The 1D CNN can extract features from local original time series data, and then model the short-term correlation between local time series data and subsequent trends [39]. So we will use the 1D CNN to analyze our data. We built a one-dimensional convolutional layer and added it to our neural network. We also add padding, which maintains the boundary information of the time series. If there is no padding, most of the obtained information will only be operated by the convolution kernel once, but the data in the middle of the sequence are scanned many times, thus the results obtained will lose the accuracy of the boundary information. To improve accuracy, we apply a casual strategy for padding, which simply pads the layer's input with zeros in the front so that we can also predict the values of early time steps in the frame [40]. Finally, we adopt Rectified Linear Unit (ReLU) as the activation function of the 1D CNN. After introducing the convolutional layer, the GRU-based layer can be added.

GRU is a derived version of RNN. RNN uses traditional backpropagation and gradient descent algorithms to learn the target data. The BackPropagation Through Time (BPTT) algorithm is a commonly used method of training RNN. The idea of BPTT is the same as the backpropagation algorithm, which continuously finds better points along the negative gradient direction of the parameters to be optimized until convergence. However, the application of the BPTT algorithm can lead to the accumulation of activation function derivatives, which in turn leads to the occurrence

of gradient disappearance and gradient explosion. In order to solve this problem, we can use two methods to avoid gradient explosion/disappearance. The first method is to replace the activation function. In our model, we avoid the disappearance of the gradient to a certain extent by setting ReLU as the activation function. But the derivative of ReLU in the range greater than 0 is always 1, which is easy to cause gradient explosion. Therefore, the second method is applied to change the circulation structure. GRU that combines the forget gate and input gate into a single "update gate" is exploited in our model. It also merges cell state and hidden state and makes some other changes.

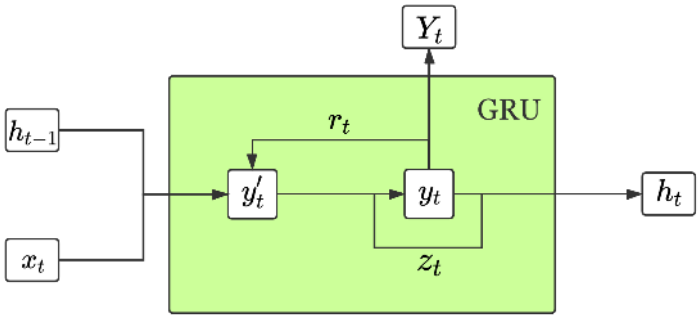


Fig. 4. GRU structure

We have adopted the GRU-based neural network and made some improvements to optimize its performance in long sequence prediction. The structure of GRU is demonstrated in Fig. 4. The reset gate r_t and update gate z_t are the same as LSTM. But there is no output gate in GRU. Compared with LSTM, there is one less "gating" inside the GRU, which has fewer parameters than LSTM, but it can also achieve functions equivalent to LSTM [41].

The sparse processing provided by ReLU can reduce the effective capacity of the model, which means too much feature masking makes the model unable to learn effective features. Since the gradient of ReLU is 0 when $x < 0$, this neuron may never be activated by any data, which is called neuron necrosis. In addition, one of the similarities between ReLU and Sigmoid is that the result is a positive value without a negative value. To address this issue, we multiply ReLU and Sigmoid, and we can get the activation function Swish that is represented as below:

$$f(x) = x \cdot \text{sigmoid}(\beta x), \tag{3}$$

where β is either a constant or a trainable parameter [42].

In the choice of activation function, instead of choosing the ReLU activation function that is commonly used by DNNs, we use Swish, which is a smooth and non-monotonic function. Its design is inspired by the use of sigmoid function for gating in LSTM. We use the same value for gating to simplify the gating mechanism, which is called self-gating. The advantage of self-gating is that it only requires a simple scalar input, while traditional gating requires multiple scalar inputs. This feature allows Swish to easily replace activation functions that take a single scalar as input without changing the hidden capacity or number of parameters. The pseudocode of esDNN is shown in Algorithm 2.

Algorithm complexity analysis: The time complexity of esDNN depends on the number of networks (N), number of network weight connections (C), number of input node (n), hidden nodes (h), where $h \approx n$, dropout value (d). Therefore, the total time complexity for a maximum number of b iterations is represented as $b \times O(n^2 \times N \times C \times d)$, which equals to $O(n^2bdNC)$.

Algorithm 2: efficient supervise learning-based Deep Neural Network (esDNN)

Input : Multivariate time series dataset R_n.
Hyperparameter:time sequence t, training epochs b
After processing by the S-MTF algorithm, multivariate time series dataset R_n is transformed into the supervised learning dataset S_n
Divide S_n into training set Ts and validation set Vs
esDNN: Conv1D(filters=32, kernel_size=5)
for i *range from* 0 *to* t **do**
    GRU:
    Update the reset gate r_t:
    $r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$
    Update the update gate z_t:
    $z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$
    Calculate the candidate hidden layer y'_t:
    $y'_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$
    Compute the output gate y_t:
    $y_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$
end
Dense(16, activation="swish")
Dropout(0.2)
Dense(1)
for j *range from* 0 *to* b **do**
    Train Ts with esDNN, compare with Vs
end

5 PERFORMANCE EVALUATION

In this section, we will firstly introduce the details about the dataset we use and the experimental configurations for workload prediction. Then we compare the performance of esDNN and other RNN-based approaches. Finally, we demonstrate that our approach can be applied to auto-scaling for cloud resource provisioning optimization.

5.1 Datasets and Environment Configuration

We implement the multivariate time series forecasting based on TensorFlow 2.2.0 [43], and the Python version is 3.7. We used two real-world datasets in the experiments for performance evaluation of our proposed approach.

- **Alibaba dataset** [8]: It is cluster-trace-v2018 of Alibaba that recording the traces in 2018. Cluster-trace-v2018 includes about 4,000 machines in a period of 8 days, which we use all the data to make predictions. The data can be found from Github[1].
- **Google dataset** [44]: It is derived from Google's cluster data-2011-2 recorded in 2011. The cluster data-2011-2 trace includes 29 days of data that contains 37,747 machines, including three different machine types. The data can also be fetched from Github[2].

Both of these datasets can represent random features of cloud workloads. We use CPU usage as a key performance measurement of the accuracy of our prediction model. To show the effectiveness of prediction and remove the redundancy information, we configure the prediction time interval as 5 minutes. For the metadata for prediction, there are some differences between the two datasets because of the different types of data collected. For the Alibaba dataset, in addition to the time series

[1]https://github.com/alibaba/clusterdata
[2]https://github.com/google/cluster-data

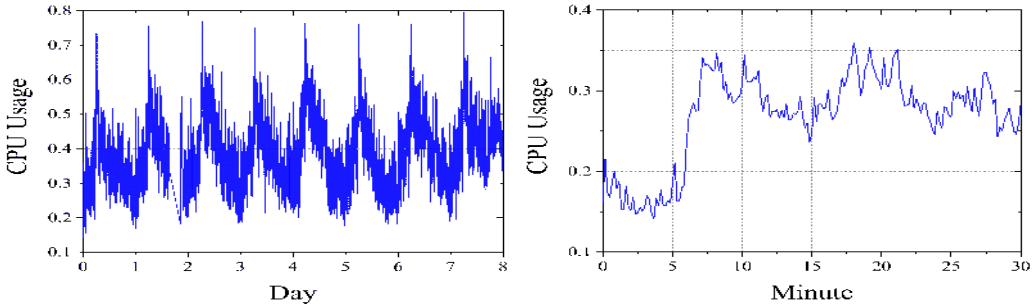


Fig. 5. Alibaba (a) per-day workload (b) per-minute workload fluctuations

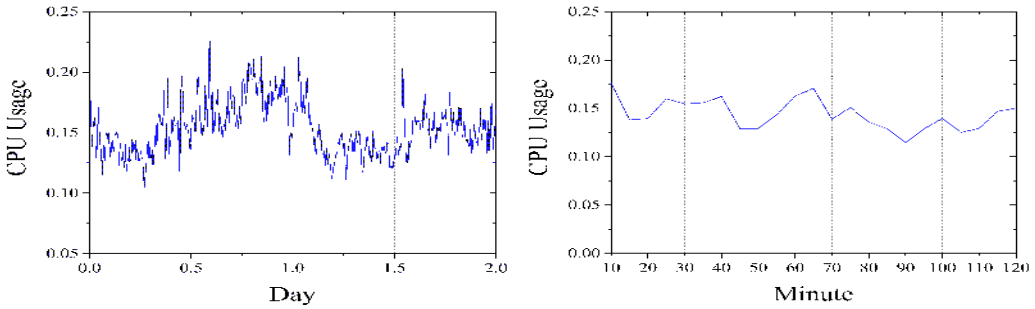


Fig. 6. Google (a) per-day workload (b) per-minute workload fluctuations

and CPU usage data, we also select memory usage, incoming network traffic, outgoing network traffic, and disk I/O usage as the source data for prediction. As for Google dataset, in addition to the time series and CPU usage data, we also select canonical memory usage, assigned memory usage, total page cache memory usage as the source data for prediction. When processing the dataset from Google, we select 5 minutes as the time interval. Then we group the tasks according to the Machine ID and finally normalize the dataset.

Figs. 5 and 6 demonstrate the CPU usage in the datasets of Alibaba and Google cloud data centers, respectively. We have divided them into per-day and per-minute workloads fluctuations of machines so that we can see the fluctuations of CPU usage more clearly over time. We can notice that both the datasets show high variance and random features. For the Alibaba dataset, we divide the dataset into the first 40,000 rows of data (59.5%) and the rest, which are used to train and test the model. Similarly, we have divided the Google dataset in this way. We selected about 72 hours of data from Google's dataset, and we used the first 49 hours (68.4%) as the training set and the rest as the validation set. For these two datasets, the number of training epoch is 200, the batch size is 72, the loss function is Huber, the optimizer is Adam, and the metric we use is mean square errors.

5.2 Comparison with Unsupervised Learning-based Approach

In contrast to the supervised learning approach used by esDNN, the unsupervised learning approach can also be applied to high-dimensional problems such as multivariate time series forecasting, therefore, in this section, we evaluate our approach with unsupervised learning-based approach.

Among the unsupervised learning approaches, Autoencoder is a representative one for efficient feature extraction and feature representation of high-dimensional data [45]. Currently, Autoencoder as well as Stacked Autoencoder, Sparse Autoencoder [46], and Denoising Autoencoder [47] are widely used in the research field. Autoencoder maps the input sample x to the hidden layer by the encoder (g) and then maps it back to the original space by the decoder (f) to obtain the reconstructed sample. For the neural network-based autoencoder model, the encoder part compresses the data by reducing the number of neurons layer by layer, while the decoder part improves the number of neurons layer by layer based on the abstract representation of the data, and finally realizes the reconstruction of the input samples. The optimization objective is to optimize both the encoder and decoder by minimizing the Loss function. The optimization equation is shown as below:

$$f, g = \min_{f,g} Loss(x, f(g(x))). \tag{4}$$

The prediction results of esDNN and Autoencoder within 10 minutes are shown in Fig. 7, which demonstrates that Autoencoder has a good prediction result only at the beginning of the observed period, and it is significantly less accurate than esDNN. The reason can be that Autoencoder does not need to use the label of the sample in prediction, and it uses the input of the sample as both the input and output of the neural network. Although this can greatly improve the generality of the model, autoencoder is prone to be overfitting when the parameters of the neural network are complicated. Based on the results compared with unsupervised learning, supervised learning based approach has demonstrated better performance. In the following experiments, we evaluate the performance with other neural network-based approaches.

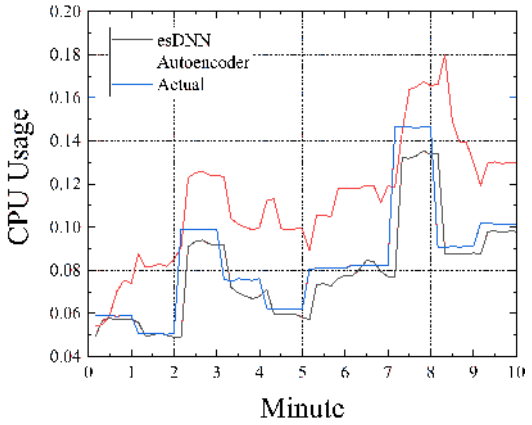


Fig. 7. Comparison of the prediction results of esDNN and Autoencoder

5.3 Comparison with Neural Network-based Approaches

We first start the evaluation with the Alibaba dataset. To compare with our algorithm, we have selected several RNN-based deep learning algorithms, which have been applied for time series prediction, including RNN, Bi-LSTM [28] and GRU. We compare the prediction accuracy of these four algorithms and measure them by Mean Square Errors (MSE), which is represented as:

$$MSE = \frac{1}{m} \sum_{i=1}^{m} \left(y_{actual} - y_{predict}\right)^2, \tag{5}$$

where m represents the timestamp, y_{actual} is the actual value and $y_{predict}$ is the predictive value. The higher the MSE of the algorithm, the greater the gap between the predicted value and the actual value. In order to capture the changing trend of MSE in each period, we set four different time scales: second, minute, hour and day.

Fig. 8 shows the MSE fluctuation of these four RNN-based methods based on the Alibaba dataset with various prediction lengths. In general, all the MSE curves follow the same trend, which shows that the MSE value first increases until it reaches a peak, after that, the curves will remain at a relatively stable value. For the second-level prediction, apart from keeping RNN at a relatively high value, there are just subtle differences between Bi-LSTM, GRU and esDNN. With the increase of the prediction length, all these curves are maintained at relatively stable values. But for the day-level prediction, there is no significant difference in MSE value between Bi-LSTM, GRU and esDNN. The main reason is that the RNN and Bi-LSTM models are designed to process time-series data and can perform well in representing the nonlinear relationship between data and time. However, the drawback of RNN and Bi-LSTM is that their gradient can disappear or explode, especially for the long time-series data during the data training process. The GRU can alleviate the side effects of gradient disappearance that usually happens in Bi-LSTM and RNN. Therefore, the results of GRU-based approach can maintain relatively more stable values compared with RNN and Bi-LSTM.

In order to brighten the differences between them, we choose to use the Cumulative Distribution Function (CDF) method to measure them, which is the integral of the probability density function. For discrete variables, it can represent the sum of the probability of occurrence of all values less than or equal to x, which is formulated as:

$$F_X(x) = \mathrm{P}(X \leq x). \tag{6}$$

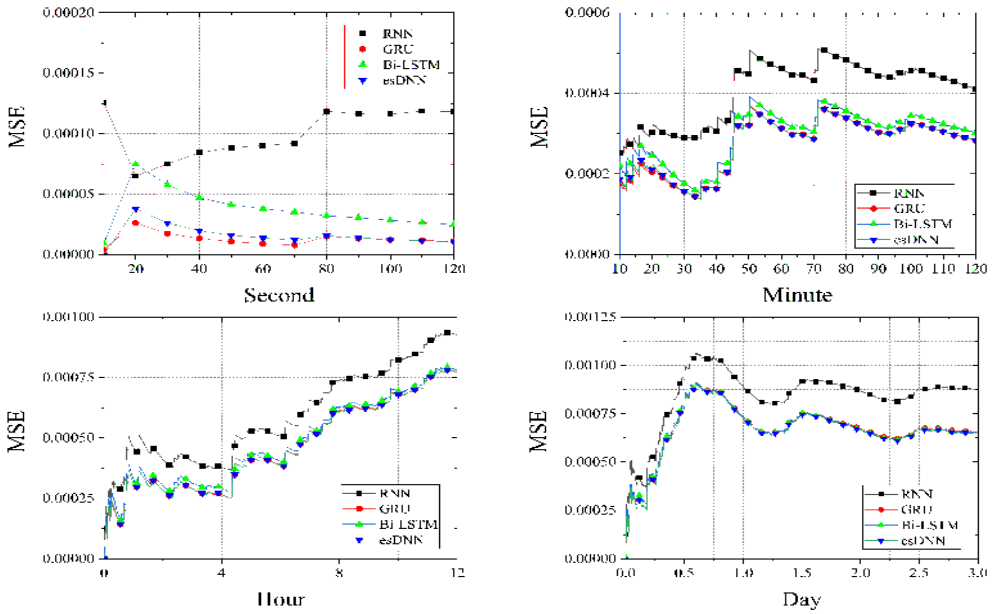
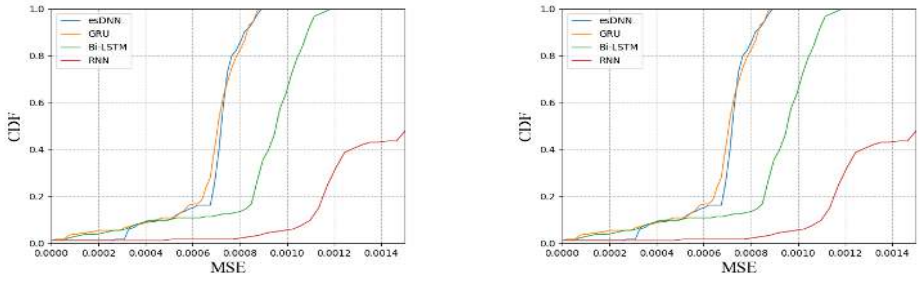


Fig. 8. Prediction accuracy (MSE) of four different RNN-based methods based on the Alibaba dataset with different prediction lengths



(a) CDF of MSE with Alibaba Trace (b) CDF of MSE with Google Trace

Fig. 9. CDF comparison of MSE curve

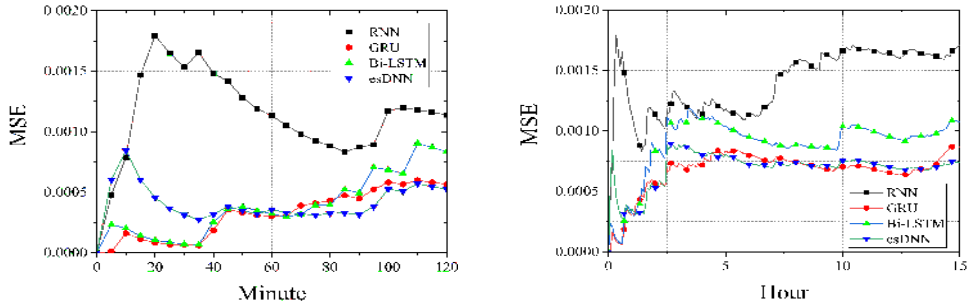
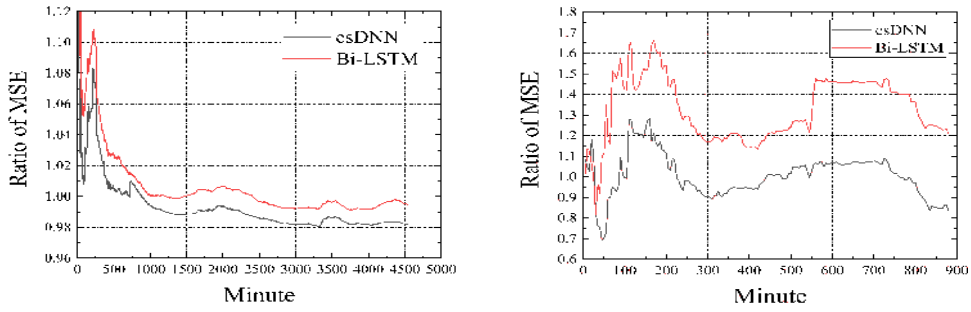


Fig. 10. Prediction accuracy (MSE) of four different RNN methods based on the Google dataset

Fig. 9 shows the CDF of MSE based on these four RNN-based methods, these almost overlapping curves in Fig. 8 can be distinguished more apparently by the difference in CDF values. Since RNN is quite different from the other three methods, and its effect is the worst one, therefore, we focus on the discussions on the other three algorithms. Except for the RNN method, we can clearly see that when the value of MSE is between 0.006 and 0.008, the value of CDF rises very quickly, which shows that the MSE values of these three methods are concentrated. Meanwhile, esDNN rises significantly faster than Bi-LSTM and GRU, we can see that for any MSE in this time period, the CDF value of esDNN always remains at the highest value. Although the curves are very close, the difference in value between them can still be easily identified. It means that the overall MSE value of esDNN is smaller than the values of Bi-LSTM and GRU.

For Google's dataset, we also use the RNN-based methods as baselines for esDNN. Compared with the Alibaba dataset, we utilize less data from Google's dataset, therefore, we show the prediction length with minute-level and hour-level. Fig. 10 shows the MSE fluctuation of the four methods based on the Google dataset. For the minute-level prediction, all these methods have little difference between each other except RNN. When we focus on the hour-level prediction, the trend of these methods is stable after a short growth, which is consistent with the results of the Alibaba dataset. We can also notice that RNN always maintains at a high level. Compared with GRU and esDNN, Bi-LSTM has a higher MSE. For esDNN and GRU, the MSE of these two are quite close, where the esDNN can achieve a more stable trend, while GRU fluctuates more dynamically. To conclude, we can notice that the prediction result of esDNN is better than GRU, since the MSE value of esDNN is smaller than GRU.



(a) Ratio of MSE with Alibaba Trace (b) Ratio of MSE with Google Trace

Fig. 11. The ratio of MSE compared with GRU for Alibaba and Google Traces

We notice that esDNN is very close to GRU's results over a long sequence of time while the trend of RNN is much worse than the other three approaches. Thus we choose to compare the MSE values without RNN. We analyze the MSE values from Alibaba and Google dataset separately, as shown in Fig. 11, where the GRU is set as the baseline, and the MSE values of esDNN and Bi-LSTM are divided by the MSE values of GRU. The value less than 1.0 represents better performance than GRU, and vice versa. Although GRU performs better prediction results than esDNN in a short period of time, esDNN can maintain better accuracy and stability in the long run.

Next, we evaluate the difference between the predicted value and the actual value of esDNN. Figs. 12 and 13 show the CPU usage curves, so that we can see the difference between the predicted value and the actual value. For the analysis of the Alibaba dataset, we can observe that for the minute level prediction, though there are some large differences between consecutive values, esDNN can still give relatively accurate prediction results. For the hour-level prediction, on the whole, the predicted value is very close to the actual value because their curves are almost fit, and only a small part of the predicted curve is different from the actual value. For hour-level prediction based on Google cluster data, esDNN can still accurately predict the trend of CPU usage.

In order to identify the difference between them more intuitively, we summarize the performance of these algorithms as listed in Table 6 and Table 7. Apart from MSE, we also compare Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) that have been widely used to evaluate prediction performance. The esDNN approach can achieve the lowest MSE values compared with other baselines when the prediction length is longer, which is more difficult to be predicted. For the Google dataset, our approach can also achieve the lowest RMSE with a longer prediction length. The reason is that our proposed prediction approach based on revised GRU and CNN not only captures the periodical features inherent in the data, but also significantly reduces the impact of resource variations on prediction results.

To compare the performance of different approaches in terms of training and predicting cost, we compare the training time and prediction time as shown in Table 8. The training time is the average time consumed for training one epoch, and the prediction time is the mean value of predicting 1000 lines of data by repeating 10 times. Based on the results, we can observe that the esDNN consumes the longest training time with about 10% more time than Bi-LSTM and GRU, which is an acceptable cost considering the performance improvement in prediction accuracy. The longer training time can result from the more complicated model of esDNN with application of CNN. And for the prediction time, esDNN can perform slightly better than Bi-LSTM and GRU, the reason can be the Swish activation function that we use can slightly improve the prediction time [48].

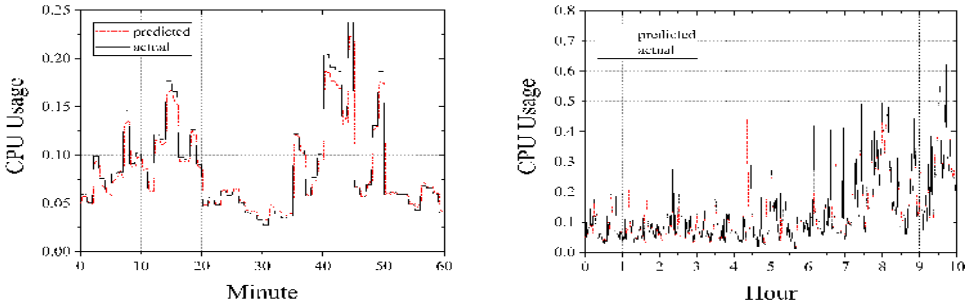


Fig. 12. Prediction performance of the esDNN compared with actual Alibaba data

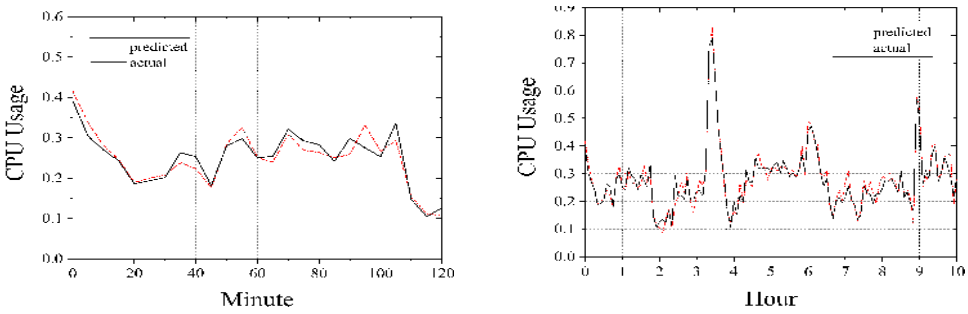


Fig. 13. Prediction performance of the esDNN compared with actual Google data

Table 6. MSE, RMSE and MAPE comparison with Alibaba dataset

Prediction length	RNN			GRU			Bi-LSTM			esDNN		
	MSE	RMSE	MAPE	MSE	RMSE	MAPE	MSE	RMSE	MAPE	MSE	RMSE	MAPE
10s	1.25E-04	0.0112	0.2175	4.27E-06	0.0021	0.0401	9.33E-06	0.0031	0.1596	**1.59E-07**	**0.0004**	**0.0077**
30s	7.50E-05	0.0087	0.1396	**1.75E-05**	**0.0042**	**0.0541**	5.77E-05	0.0076	0.1086	3.13E-05	0.0056	0.0681
1 min	9.02E-05	0.0095	0.1568	**9.02E-06**	**0.0030**	**0.0325**	3.80E-05	0.0062	0.0891	1.71E-05	0.0041	0.0480
30 min	2.93E-04	0.0170	0.2024	**1.71E-04**	**0.0125**	0.0772	2.00E-04	0.0142	0.0978	1.84E-04	0.0128	**0.0766**
1h	4.61E-04	0.0215	0.2343	**3.13E-04**	**0.0177**	0.0936	3.31E-04	0.0182	0.1053	3.20E-04	0.0179	**0.0891**
6h	5.07E-04	0.0225	0.2237	**3.88E-04**	**0.0197**	0.1050	4.03E-04	0.0200	**0.0891**	3.96E-04	0.0199	0.0990
1 day	8.91E-04	0.0293	0.2387	7.34E-04	0.0271	0.1260	7.32E-04	0.0271	**0.0898**	**7.25E-04**	**0.0269**	0.1080
2 days	8.68E-04	0.0295	0.2058	6.74E-04	0.0260	0.1050	6.66E-04	0.0258	**0.0872**	**6.62E-04**	**0.0257**	0.0901
3 days	8.83E-04	0.0297	0.1973	6.54E-04	0.0256	0.0978	6.50E-04	0.0255	0.0857	**6.43E-04**	**0.0254**	**0.0842**

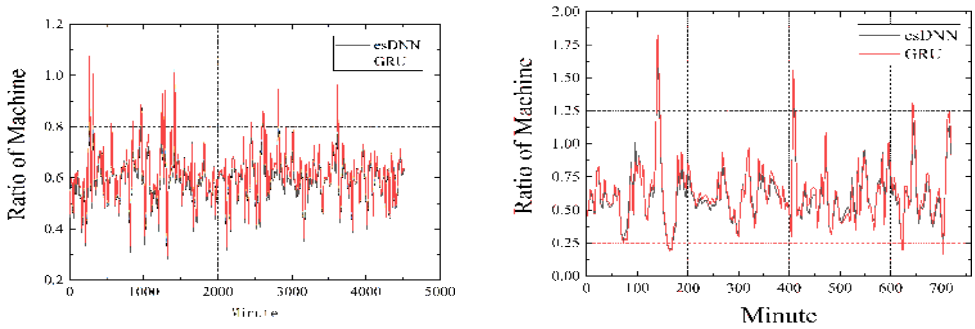
Table 7. MSE, RMSE and MAPE comparison with Google dataset

Prediction length	RNN			GRU			Bi-LSTM			esDNN		
	MSE	RMSE	MAPE	MSE	RMSE	MAPE	MSE	RMSE	MAPE	MSE	RMSE	MAPE
30min	0.00153232	0.0391	0.1651	**6.53E-05**	**0.0081**	0.0228	7.22E-05	0.0085	**0.0209**	0.00031276	0.0177	0.0439
1h	0.0011334	0.0337	0.1201	**0.00030235**	**0.0174**	**0.0506**	0.00032074	0.0179	0.0567	0.0003576	0.0189	0.0507
2h	0.00113796	0.0337	0.1173	0.00056542	0.0238	0.0737	0.00082948	0.0288	0.0847	**0.00052629**	**0.0229**	**0.0691**
4h	0.00113856	0.0337	0.1246	**0.00071719**	**0.0268**	0.0942	0.0011012	0.0332	0.1143	0.00079793	0.0282	**0.0884**
6h	0.00113166	0.0336	0.1087	0.00079771	0.0282	0.0886	0.00094857	0.0308	0.1010	**0.0007228**	**0.0269**	**0.0787**
8h	0.00156561	0.0396	0.1350	0.00074263	0.0273	0.0880	0.00086032	0.0298	0.0960	**0.00073787**	**0.0272**	**0.0833**
12h	0.00164631	0.0406	0.1368	**0.00065455**	**0.0256**	0.0819	0.00095340	0.0309	0.0954	0.00070197	0.0265	**0.0814**
15h	0.00170941	0.0413	0.1368	0.00086864	0.0295	0.0876	0.0010403	0.0325	0.1017	**0.00073697**	**0.0271**	**0.0822**

To summarize, esDNN can achieve good accuracy based on MSE results. Compared with the MSE results evaluated in the same datasets derived from Google and Alibaba in [7], esDNN has reduced the MSE with one order of magnitude from around 7×10^{-2} to 7×10^{-3}.

Table 8. Training time and prediction time comparison

	RNN	Bi-LSTM	GRU	esDNN
Traing time (s)	5.11	6.47	6.53	7.14
Prediction time (s)	0.048	0.070	0.071	0.065



(a) Optimization ratio of Alibaba machines (b) Optimization ratio of Google machine

Fig. 14. Optimization ratio comparison in minutes with different traces

5.4 Applying esDNN for Machines Auto-scaling with Simulations

The auto-scaling technique can dynamically adjust the number of active machines in the system based on the system status, e.g. removing machines when the system is at a low utilization level or adding more machines when the system is overutilized. By taking advantage of auto-scaling, the system performance, e.g. energy consumption, can be optimized. However, without sufficiently accurate prediction for workloads, the popular threshold-based auto-scaling approaches, like static threshold, are undesirable for workloads with high variance.

To further demonstrate the capability of the proposed approach, we integrate esDNN into the auto-scaling scenario for physical machines in Alibaba and Google cloud data centers by simulating the number of machines and resource usage. The specifications of machines are derived from the corresponding original datasets, and the scheduling period is configured as 5 minutes.

Our objective is to improve resource utilization and reduce the number of active machines with sufficient accurate predictions. Therefore, the use of CPU utilization as an input to the auto-scaling method is highly desired. And the output is the number of active machines. As an auto-scaling baseline, we use the average number of active machines based on the previous time slots [49], which can be calculated as:

$$M(t) = \frac{\sum_{i=1}^{m} M(t-i)}{m}, \tag{7}$$

where $M(t)$ represents the number of active machines at time interval t, and m is the number of previous time slots used for the prediction that we set m as 5 for our experiments. The number of active machines calculated by Eq. 7 is normalized as 1.0. We normalize the number of active machines of the auto-scaling approach based on esDNN and calculate the ratio between the esDNN-based approach and baseline, and we configure the upper CPU utilization threshold as 80%. If the ratio of machines is less than 1.0, it means that the esDNN-based approach can reduce the number of active machines. Fig. 14a shows the prediction of the number of active machines based on the Alibaba dataset. As expected, the ratio fluctuates in the range of 0.3 to 0.6, indicating that our prediction algorithm has achieved a good effect that only 40% to 80% of the number of machines will

be active compared with the original number of machines from the dataset, which can significantly reduce the number of active machines.

As for the Google dataset, we analyze the capacity distribution of about 37,678 machines, where the machines with 0.5 capacity are about 92.8%, the machines with 0.25 capacity are about 1.4%, and the machines with capacity 1.0 are about 5.9%. This is different from the distribution of the homogeneous machines in the Alibaba dataset. After normalizing the CPU usage in the Google dataset, we also utilize the algorithm previously applied to the Alibaba dataset, and Fig. 14b shows the ratio ranges from 0.25 to 1. For example, at the 400th minute, the baseline needs 18,371 machines, while our approach only uses 5,161 machines. The results of these experiments are close to those based on the Alibaba dataset. It can be concluded that the proposed approach can efficiently improve resource usage by reducing the number of active machines, and it is promising to reduce the energy consumption of cloud data centers by providing an accurate prediction method.

6 CONCLUSIONS AND FUTURE WORK

Our deep learning-based approach for cloud workload prediction brings opportunities to optimize resource provisioning in the cloud computing environment. In this paper, we apply sliding window for multivariate time series to convert the high dimension data into supervised learning time series to address the high dimensionality challenge. Based on the converted data, we proposed a revised GRU-based approach to train the prediction model to achieve high prediction accuracy for high variance cloud workloads. Comprehensive experiments based on the realistic traces derived from Google and Alibaba have demonstrated that our proposed approach can achieve better performance in terms of accuracy compared with state-of-the-art approaches. To further show the effectiveness of optimizing resource provisioning, we applied our approach for auto-scaling based on realistic traces, and results illustrate that our approach can significantly optimize the resource usage of cloud data centers, thus saving operational costs.

In future, our approach can be integrated into a container-based prototype system, e.g. Kubernetes, to optimize resource provisioning. We would like to investigate the proposed approach to be extended for Edge Computing to reduce response time using offloading techniques, and consider the location-aware and mobility-aware scenarios (e.g. predicting the loads allocating to different devices generated by mobile users). We would also like to make automatic esDNN by using Monitor, Analyze, Plan, and Execute (MAPE) model.

ACKNOWLEDGMENT

This work is supported by National Key R&D Program of China (No. 2021YFB3300200), National Natural Science Foundation of China (NO. 62102408 and 62071327), Shenzhen Basic Research Program (No. JCYJ20200109115418592), and Youth Innovation Promotion Association CAS (2019349).

REFERENCES

[1] Minxian Xu and Rajkumar Buyya. Brownout approach for adaptive management of resources and applications in cloud computing systems: A taxonomy and future directions. *ACM Computing Surveys*, 52(1):8:1–8:27, 2019.

[2] Rajkumar Buyya, Chee Shin Yeo, and Srikumar Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications*, pages 5–13, 2008.

[3] Mohammad Hossein Ghahramani, MengChu Zhou, and Chi Tin Hon. Toward cloud computing qos architecture: Analysis of cloud systems and cloud services. *IEEE/CAA Journal of Automatica Sinica*, 4(1):6–18, 2017.

[4] M. Du, Y. Wang, K. Ye, and C. Xu. Algorithmics of cost-driven computation offloading in the edge-cloud environment. *IEEE Transactions on Computers*, 69(10):1519–1532, 2020.

[5] Yujun Chen, Xian Yang, Qingwei Lin, Hongyu Zhang, Feng Gao, Zhangwei Xu, Yingnong Dang, Dongmei Zhang, Hang Dong, Yong Xu, Hao Li, and Yu Kang. Outage prediction and diagnosis for cloud service systems. In *The World Wide Web Conference*, WWW '19, pages 2659–2665, New York, NY, USA, 2019.

[6] S. Wang, X. Li, and R. Ruiz. Performance analysis for heterogeneous cloud servers using queueing theory. *IEEE Transactions on Computers*, 69(4):563–576, 2020.

[7] Z. Chen, J. Hu, G. Min, A. Y. Zomaya, and T. El-Ghazawi. Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning. *IEEE Transactions on Parallel and Distributed Systems*, 31(4):923–934, April 2020.

[8] W. Chen, K. Ye, Y. Wang, G. Xu, and C. 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)*, pages 102–109, 2018.

[9] T. Mikolov, S. Kombrink, L. Burget, J. Černocký, and S. Khudanpur. Extensions of recurrent neural network language model. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5528–5531, 2011.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[11] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning,*, December 2014.

[12] Hervé Abdi and Lynne J. Williams. Principal component analysis. *WIREs Computational Statistics*, 2(4):433–459, 2010.

[13] Yasi Wang, Hongxun Yao, and Sicheng Zhao. Auto-encoder based dimensionality reduction. *Neurocomputing*, 184:232 – 242, 2016. RoLoD: Robust Local Descriptors for Computer Vision 2014.

[14] Jason Brownlee. Supervised and unsupervised machine learning algorithms. *Machine Learning Mastery*, 16(03), 2016.

[15] 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*, 3(4):449–458, Oct 2015.

[16] Jingqi Yang, Chuanchang Liu, Yanlei Shang, Bo Cheng, Zexiang Mao, Chunhong Liu, Lisha Niu, and Junliang Chen. A cost-aware auto-scaling approach using the workload prediction in service clouds. *Information Systems Frontiers*, 16(1):7–18, March 2014.

[17] Katja Cetinski and Matjaz B. Juric. Ame-wpc: Advanced model for efficient workload prediction in the cloud. *Journal of Network and Computer Applications*, 55:191 – 201, 2015.

[18] Parminder Singh, Pooja Gupta, and Kiran Jyoti. Tasm: Technocrat arima and svr model for workload prediction of web applications in cloud. *Cluster Computing*, 22(2):619–633, June 2019.

[19] Chunhong Liu, Chuanchang Liu, Yanlei Shang, Shiping Chen, Bo Cheng, and Junliang Chen. An adaptive prediction approach based on workload pattern discrimination in the cloud. *Journal of Network and Computer Applications*, 80:35 – 44, 2017.

[20] J. Bi, H. Yuan, and M. Zhou. Temporal prediction of multiapplication consolidated workloads in distributed clouds. *IEEE Transactions on Automation Science and Engineering*, 16(4):1763–1773, 2019.

[21] Jitendra Kumar and Ashutosh Kumar Singh. Workload prediction in cloud using artificial neural network and adaptive differential evolution. *Future Generation Computer Systems*, 81:41 – 52, 2018.

[22] 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*, 14(7):3170–3178, 2018.

[23] Jitendra Kumar, Rimsha Goomer, and Ashutosh Kumar Singh. Long short term memory recurrent neural network (lstm-rnn) based workload forecasting model for cloud datacenters. *Procedia Computer Science*, 125:676 – 682, 2018.

[24] F. Qiu, B. Zhang, and J. Guo. A deep learning approach for vm workload prediction in the cloud. In *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 319–324, May 2016.

[25] Yonghua Zhu, Weilin Zhang, Yihai Chen, and Honghao Gao. A novel approach to workload prediction using attention-based lstm encoder-decoder network in cloud environment. *EURASIP Journal on Wireless Communications and Networking*, 2019(1):274, 2019.

[26] Maryam Amiri, Leyli Mohammad-Khanli, and Raffaela Mirandola. An online learning model based on episode mining for workload prediction in cloud. *Future Generation Computer Systems*, 87:83 – 101, 2018.

[27] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, page 153–167, New York, NY, USA, 2017.

[28] Jing Bi, Shuang Li, Haitao Yuan, and MengChu Zhou. Integrated deep learning method for workload and resource prediction in cloud systems. *Neurocomputing*, 424:35–48, 2021.

[29] Md Ebtidaul Karim, Mirza Mohd Shahriar Maswood, Sunanda Das, and Abdullah G Alharbi. Bhyprec: A novel bi-lstm based hybrid recurrent neural network model to predict the cpu workload of cloud virtual machine. *IEEE Access*, 9:131476–131495, 2021.

[30] Chuang Chen, Ningyun Lu, Bin Jiang, and Cunsong Wang. A risk-averse remaining useful life estimation for predictive maintenance. *IEEE/CAA Journal of Automatica Sinica*, 8(2):412–422, 2021.

[31] Ashutosh Kumar Singh, Deepika Saxena, Jitendra Kumar, and Vrinda Gupta. A quantum approach towards the adaptive prediction of cloud workloads. *IEEE Transactions on Parallel and Distributed Systems*, 2021.
[32] In Kee Kim, Wei Wang, Yanjun Qi, and Marty Humphrey. Forecasting cloud application workloads with cloudinsight for predictive resource management. *IEEE Transactions on Cloud Computing*, 2020.
[33] Wei Sun, Pengyu Li, Zhi Liu, Xue Xue, Qiyue Li, Haiyan Zhang, and Junbo Wang. Lstm based link quality confidence interval boundary prediction for wireless communication in smart grid. *Computing*, 103:251–269, 2021.
[34] Xuesong Li, Yating Liu, Kunfeng Wang, and Fei-Yue Wang. A recurrent attention and interaction model for pedestrian trajectory prediction. *IEEE/CAA Journal of Automatica Sinica*, 7(5):1361–1370, 2020.
[35] Silvio Barra, Salvatore Mario Carta, Andrea Corriga, Alessandro Sebastian Podda, and Diego Reforgiato Recupero. Deep learning and time series-to-image encoding for financial forecasting. *IEEE/CAA Journal of Automatica Sinica*, 7(3):683–692, 2020.
[36] Junfei Qiao, Fei Li, Cuili Yang, Wenjing Li, and Ke Gu. A self-organizing rbf neural network based on distance concentration immune algorithm. *IEEE/CAA Journal of Automatica Sinica*, 7(1):276–291, 2019.
[37] Thomas G Dietterich. Machine learning for sequential data: A review. In *Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)*, pages 15–30. Springer, 2002.
[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
[39] Dongkuan Xu, Wei Cheng, Bo Zong, Dongjin Song, Jingchao Ni, Wenchao Yu, Yanchi Liu, Haifeng Chen, and Xiang Zhang. Tensorized lstm with adaptive shared memory for learning trends in multivariate time series. In *AAAI*, pages 1395–1402, 2020.
[40] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.
[41] Rui Fu, Zuo Zhang, and Li Li. Using lstm and gru neural network methods for traffic flow prediction. In *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pages 324–328. IEEE, 2016.
[42] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.
[43] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, Savannah, GA, November 2016.
[44] John Wilkes. More Google cluster data. Google research blog, November 2011. Posted at http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html.
[45] Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4):291–294, 1988.
[46] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.
[47] Xugang Lu, Yu Tsao, Shigeki Matsuda, and Chiori Hori. Speech enhancement based on deep denoising autoencoder. In *Interspeech*, volume 2013, pages 436–440, 2013.
[48] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *CoRR*, abs/1710.05941, 2017.
[49] Mohammad Sadegh Aslanpour, Mostafa Ghobaei-Arani, and Adel Nadjaran Toosi. Auto-scaling web applications in clouds: A cost-aware approach. *Journal of Network and Computer Applications*, 95:26 – 41, 2017.