J Grid Computing (2022) 20: 16
https://doi.org/10.1007/s10723-022-09607-0

Workload Time Series Cumulative Prediction Mechanism for Cloud Resources Using Neural Machine Translation Technique

Mustafa M. Al-Sayed

Received: 6 September 2021 / Accepted: 17 April 2022 / Published online: 4 May 2022

Abstract Dynamic resource allocation and auto-scaling represent effective solutions for many cloud challenges, such as over-provisioning (i.e., energy-wasting, and Service level Agreement "SLA" violation) and under-provisioning (i.e., Quality of Service "QoS" dropping) of resources. Early workload prediction techniques play an important role in the success of these solutions. Unfortunately, no prediction technique is perfect and suitable enough for most workloads, particularly in cloud environments. Statistical and machine learning techniques may not be appropriate for predicting workloads, due to instability and dependency of cloud resources' workloads. Although Recurrent Neural Network (RNN) deep learning technique considers these shortcomings, it provides poor results for long-term prediction. On the other hand, Sequence-to-Sequence neural machine translation technique (Seq2Seq) is effectively used for translating long texts. In this paper, workload sequence prediction is treated as a translation problem. Therefore, an Attention Seq2Seq-based technique is proposed for predicting cloud resources' workloads. To validate the proposed technique, real-world dataset collected from a Google cluster of 11 k machines is used. For improving the performance of the proposed technique, a novel procedure called cumulative-validation is proposed as an alternative procedure to cross-validation. Results show the effectiveness of the proposed technique for predicting workloads of cloud resources in terms of accuracy by 98.1% compared to 91% and 85% for other sequence-based techniques, i.e. Continuous Time Markov Chain based models and Long short-term memory based models, respectively. Also, the proposed cumulative-validation procedure achieves a computational time superiority of 57% less compared to the cross-validation with a slight variation of 0.006 in prediction accuracy.

Keywords Cloud computing · RNN · Sequence-to-sequence · Workload prediction · Neural Machin translation · Attention

M. M. Al-Sayed (✉)
Department of Computer Science, Faculty of Computers and Information, Minia University, Minya, Egypt
e-mail: mustafa.al-sayed@mu.edu.eg
e-mail: mostafamcs@gmail.com

1 Introduction

Many organizations and individuals are being motivated to the cloud computing technology due to its great properties [1–3]. For attracting the largest number of customers, the competition is heating up among cloud providers in terms of the quality of the provisioned services (QoS) and the adherence of Service Level Agreement (SLA) [4, 5]. Therefore, each provider tries to improve the nonfunctional features, such as scalability, elasticity, availability, reliability, performance, and low cost, of cloud services provisioned to customers with maintaining a low level of power consumption [6–8]. However, improving these features has some challenges, such as dynamic resource allocation, and auto-scaling [4, 6, 8]. Early prediction of cloud resources' workloads can provide a magnificent solution

to these challenges [9]. This, in turn, will maximize profits of cloud providers [10]. Unfortunately, no prediction technique is perfect and suitable enough for most workloads particularly in cloud environments [11]. Several cloud related workload prediction techniques have been studied in this paper to construct a more efficient technique for these environments.

Neural machine translation (NMT) techniques, such as Sequence-to-Sequence (Seq2Seq), caused an essential and unexpected shift departure of mainstream research strategies in the last few years [12]. For exploiting gains of these techniques, cloud workload prediction should be treated as a translation problem. In this paper, an Attention Seq2Seq-based technique is proposed for predicting cloud workloads. Motivations and contributions of this work are clearly discussed in the following subsections.

To evaluate the proposed workload prediction technique, the real-world dataset collected from a Google cluster of 11 k machines is used. According to the experimental results, the proposed technique has proven its effectiveness in dealing with time-series-based problems in cloud environments. Also, it outperforms other sequence-based techniques, such as Continuous Time Markov Chain (CTMC) based prediction solution proposed in [13] and the STM-based prediction techniques.

1.1 Problem Motivation

Although the widespread of cloud technology among a wide sector of IT users, many of its related challenges have not been adequately addressed in research studies yet. Some of these challenges are: 1) Dynamic Resource Allocation; allocating/deallocating cloud resources efficiently maintains the adherence of the SLAs, and avoiding QoS dropping, while maintaining the minimum operational cost [14]. 2) Efficiently Consumption of Power; the proper managing of over-provisioning/under-provisioning problems represents the main engine for addressing this challenge [9]. 3) Elasticity Management; ability of clouds to adapt to workload changes by maintaining the difference between available and requested resources represents the aim of this challenge [15]. 4) Avoiding Traffic Congestions [16]. 5) Workload Prediction; heterogeneity of resource workloads in cloud environments renders the prediction of these workloads a complex challenge [17].

Understanding behaviors of those workloads can promote management operations, such as VM scheduling, resource allocation, and power consumption management. These operations have a direct influence on many cloud features, such as power efficiency, availability, elasticity, scalability, and reliability of cloud resources (i.e., CPU, Memory, and Network bandwidth) [17]. Cloud providers always promise to meet these features. Quick planning and allocation of resources are so needed to realize these promises. Workload prediction can contribute directly or indirectly to enhancing most of those features, as well as, providing promising results in terms of efficiency, operational cost, and QoS [4, 18–20]. Workload prediction prohibits extreme and inadequate allocation of cloud resources, SLA violations, QoS dropping, as well as, reduce energy consumption by making right decisions on the new VM placement and VM migration [4, 6, 19, 21]. Therefore, the early workload prediction is extremely important for addressing the previous challenges by preventing unwanted events before happening [4, 22, 23].

For example, over-provisioning of resources (i.e., requested > available) and under-provisioning (i.e., requested < a predefined threshold of available) are two problems that represent the most common factors of the power consumption issue. The over-provisioning renders the energy wasting high dramatically, while the under-provisioning causes Quality of Service (QoS) dropping and SLA violation [6, 19]. Both problems can be avoided by the early prediction of workloads, which allows scaling up resources or migrating VMs from servers with predictably over-provisioning to servers with predictably low-provisioning [24–26].

In addition, the early prediction can save the migration time and the power consumed due to the idle servers by providing a balanced allocation of cloud resources [9]. Workload prediction is the main phase in the resource allocation process [6]. It helps to maintain a high availability rate by forecasting whether the available resources are sufficient or need to be up-scaled. This may allow cloud providers to redistribute their limited resources among customers in a way that render these resources feel to be unlimited and can be scaled at any time [27]. This, in turn, contributes to achieving the main goal of the elasticity property, which is allowing customers to scale their allocated resources anytime [28]. Also, the workload is a key input to the elasticity controllers, which have just two inputs (i.e., workloads and estimated capacity of demands). However, Scaling decisions are useless when the workload has already raised. Thus, using predicted workloads instead of the

present ones as the input seems to be useful for early successful scaling decisions [11]. Therefore, workload prediction can contribute directly or indirectly to enhancing most of the cloud computing properties.

Considering no prediction technique is perfect and suitable enough for most workloads particularly in cloud environments [11], several workload prediction techniques have been studied and analyzed in this paper to construct a more efficient technique for predicting workloads of cloud resources. On the other hand, the improvement rate of Machine Translation (MT) is very fast, about 3–7% every year. Nowadays, many MT techniques are readily available for translating long sentences in a quick turnaround time [29, 30]. Neural machine translation (NMT) is one of the most recent and effective MT techniques [31]. NMT techniques are adopted at faster rates than all the other existing MT techniques due to the exponential growth of the data amount that represents the main motivation of using neural networks [29, 30, 32]. As shown in Fig. 1, NMT represents a revolution in the field of MT. It caused an essential and unexpected shift departure of mainstream research strategies in the last few years [12]. NMT techniques have undergone several transformations that have pushed these techniques to entered the mature phase [12, 33]. They have already been widely adopted in the IT industry [12].

1.2 Problem Statement

In cloud computing environments, workload prediction is not a trivial process due to the instability and the complex nature of these environments [4, 20]. For example, according to [34], usage of cloud CPU resources varies from 5% to 80% during a day. Although the effectivity of workload prediction methods in the cloud environment, these methods suffer from the high variance or so-called instability over the time series [4, 5, 8]. Therefore, selecting the prediction method, which achieves accurate forecasts, represents a cloud challenge [20].

Workload prediction can be considered a time series (i.e. sequence) prediction problem, where each workload at a specific time interval related to workloads at the previous intervals [35]. Although statistic-based prediction methods (e.g., Auto-Regression [36] [36], Moving Average [37], and Auto-Regressive Integrated Moving Average [38]) achieve acceptable accuracy for predicting sequence-based problems, these methods suppose the stationary of the collected data that may contradict the dynamic nature of cloud environments [4, 35]. Recently, machine-learning prediction methods (e.g., Bayesian [39] and k-Nearest Neighbor [40]) have been used for workload prediction, where their accuracy outperformed the statistic-based methods. Unfortunately, those learning-based methods have considered workloads as independent collection of values [35].

Although the recurrent neural network (RNN) deep learning provides a solution to the time series perdition problems by considering the correlations among the collected sequence data, few attempts have provided RNN-based techniques for predicting workloads of cloud resources [35]. Additionally, these methods provide poor results for long-term predictions [5]. In this paper, a new cloud resources' workload prediction technique is proposed to overcome the drawbacks of the previously mentioned categories of workload prediction techniques (i.e., statistic-based, machine learning-based, and RNN-based techniques).

For the validation procedures, they are the processes of evaluating a trained model using a testing dataset. These procedures aim to test the generalization ability of the trained model and find the optimal training model. Therefore, they are indispensable for achieving the best prediction performance [41]. It is observed that cross-validation is the most common procedure for evaluating various machine learning and deep learning techniques. However, theoretical problems, such as instability, dependencies, evolutionary, and missing values, attached to the time-series datasets might contradict the main assumptions of the cross-validation procedure. Therefore, the resampling procedure is not suitable [42, 43]. In this paper, a new validation procedure, which considers these theoretical problems, is proposed.

1.3 Problem Solution and Contributions

A string can be considered as a series of ordered, contiguous, and dependent characters or words, while a sequence requires the order as the only condition. For example, "seq" and "sqnc" are both subsequences of the string "sequence", while "seq" is only a substring of the same string [44]. In other words, the string meets most properties of the workload datasets, such as dependence and time-ordered measurements. Therefore, considering workload prediction as a string prediction problem, e.g. translation and Question answering, might be better. Thus, the workload prediction problem can be

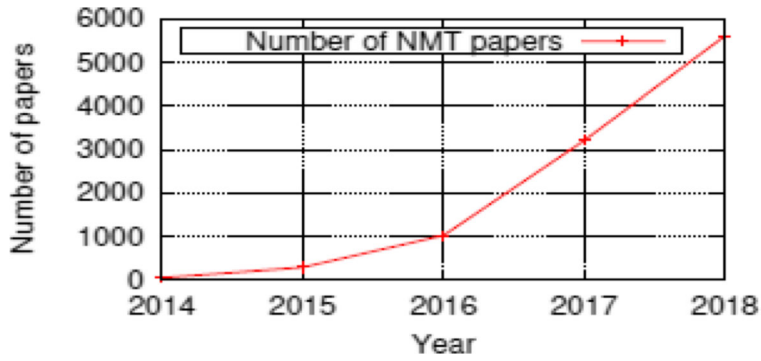

Fig. 1 Number of NMT studies in a chronological sequence according Google Scholar [12]

considered as a translation problem, where each string of measurements (i.e., word) or string of words (i.e., sentence) can be translated into its adjacently posterior word or sentence, respectively.

Long Short-Term Memory (LSTM)-based prediction models are RNN models that have not only the ability to process single data points, but also entire sequences of data [45]. LSTM model has the ability to translate long sentences correctly (i.e., effective results for long-term predictions) [46]. In 2014, Google has introduced an encoder-decoder LSTM-based RNN prediction technique called Sequence-to-Sequence (Seq2Seq). This technique provides promising results for solving the translation problem. It also allows input and output sentences to have the same or variable length [46]. Therefore, in this paper, Seq2Seq is adopted for addressing the cloud workload prediction problem.

On the other hand, Seq2Seq might fail to capture the essence of the entire sequence. This may render the prediction process of workloads goes worse as the sequence length increases [47]. For addressing this shortcoming, the attention mechanism is combined with the proposed technique.

For the validation procedure of the sequence prediction models, traditional practices are to reserve a part from end of the given sequence dataset for testing the accuracy of the constructed model, while using the rest of dataset for training [42]. Although this procedure eliminates the theoretical problems mentioned previously (i.e., dependencies, temporal evolutionary effects, and missing values) [42], it might not suitable for the nonstationary nature of workloads in cloud environments. It also takes a long computational time for the learning process, where the validation process is repeated for each fold of the dataset entirely. In this paper, a new validation procedure called **Cum**ulative-validation **P**rocedure (CumP) is proposed to address the nonstationary problem and improve the consumed time for learning. The new procedure provides a set of prediction models, one for each time interval (e.g., a month, half a month, week, half a week, or day). Each model considers measurements of its previous time intervals.

In sum, the key contributions and academic values of this work can be summarized in the following four points:

- The novelty of this study is embodied in treating the workload prediction of cloud resources problem as a machine translation problem. This might open the door to many translation techniques to be used for solving time-series prediction problems. These techniques consider long sentences and dependability among words or characters. From my point of view, this renders the process of predicting too long sequences possible. Therefore, a neural machine translation-based workload prediction technique (i.e., Attention-based Seq2Seq) is proposed.
- Workloads in cloud environments do not have a stable pattern, where customer demands are changed from time to time [9]. So, adopting a single model for predicting workloads over the whole time might not be suitable in these dynamic environments. In this paper, the whole prediction time is suggested to be divided into subintervals and constructing a specific model for each subinterval particularly. For example, constructing seven models for predicting the daily events (i.e., assigning a model for each day) might be better than a single model for the whole week. Specifying the suitable length of the learning period related to each subinterval can be considered as an additional problem that needs to be addressed in this paper.
- The majority of workload prediction studies adopt the cross-validation procedure for building and

evaluating their models. However, theoretical problems, such as instability, dependencies, evolutionary, and missing values, attached to the time-series datasets (e.g., workload datasets) might contradict the main assumptions of this procedure. In addition, the resampling process of datasets followed by that procedure renders the consumed time for building models is very long. For saving about 57% of computational time consumed due to the resampling process, as well as, considering problems attached to workload datasets, a novel cumulative-validation procedure is proposed in this paper. Suppose time-series dataset divided into k folds (f_1, $f_2 \ldots f_k$), in case of cross-validation; (f_3 to f_k & f_1) and (f_1 to f_4 & f_6 to f_k) can be used for predicting f_2 and f_5, respectively. This contradicts the basic concept of time-series problems, where f_2 depends only on f_1 and f_5 depends on f_1 to f_4. This is what has been taken into account in this paper. According to the proposed cumulative-validation procedure, time-period-based prediction models have been developed (i.e., model for each time period) for addressing instability and evolutionary problems.

– The smoothing process and discretizing techniques have a great impact on the accuracy of the constructed prediction models. In this paper, the impact of using clustering techniques (e.g., K-means clustering) as discretizing technique combined with the most common filtering techniques (i.e., Savitzky-Golay [48]) has been studied for smoothing cloud resources workloads dataset.

This paper is organized as follows; the existing workload prediction solutions for cloud resources are discussed in Section 2. An overview of the deepest learning techniques related to the proposed technique is presented in Section 3. In Section 4, the proposed workload prediction technique, as well as, the proposed validation procedure are described. The performance of the proposed technique and the proposed validation procedure is evaluated in Section 5. Finally, the conclusions and future work are presented in Section 6.

2 Related Work

Many studies, such as [4, 7, 49–54], have proposed using prediction mechanisms for managing cloud resources. These studies can be classified into three classes; deep learning-based, statistical-based, and machine learning-based studies presented in this section in two groups. The most recent and related neural network (NN) based studies are presented in the first group, while the second one includes the last two classes as there are many studies that rely on both together for proposing workload prediction solutions.

2.1 Deep Learning Based Studies

In this subsection, the most related studies to neural network based workload prediction of cloud resources are discussed in the following paragraphs.

For identifying future trends of cloud resources, acting as per workload demands, and addressing many of cloud issues, such as availability maintenance and energy consumption minimization), authors in [55] have proposed a LSTM-based deep learning model for time series prediction of cloud servers' workloads. The main drawbacks of this model can be summarized in two points: 1) its measured accuracy has not been evaluated compared to any other related time series prediction techniques. 2) The dataset used to construct that model has not been collected a real cloud computing environment.

For maintaining a more interactive cloud gaming services that satisfy users demands (e.g., quick response and streaming videos with high quality), authors in [56] have proposed a LSTM-based technique for predicting irregular player workloads. According to the predicted workloads, necessary cloud resources are scheduled to be ready for allocation using Fractional Rider-based Harmony Search Algorithm (Rider-based HSA). This algorithm is a compilation of Fractional calculus (FC), Rider optimization algorithm (ROA) and Harmony search algorithm (HSA). The main drawback of this technique can be summed up in one point, which is that source and format of the used dataset have not been mentioned clearly.

For an intelligent allocation of cloud resources, authors in [57] have proposed a deep learning based technique for the early workload prediction on these resources. This technique integrates both bi-directional and Grid LSTM neural network. To get rid of noise and reduce the standard deviation, authors have applied a set of operations at the data preparation phase, such as smoothing, filtering, and Min–Max scaling operations.

Authors, in [58], have presented a comparative practical study for analyzing the performance of using the normal neural network to predict consumption of virtual

machine resources (e.g., CPU, and memory) against the Auto-Regressive Integrated Moving Average (ARIMA) [59]. According to their experimental results, the neural-based prediction model achieves the superiority. The used dataset composed of consumptions of VM resources in two months at 15-min intervals (i.e., about 5700 readings). One of the main drawbacks of this study is the adoptance of the Bagging principle, where the used dataset is randomly splitted many times to construct different prediction models. Although the same authors ensured that the Bagging are not always achieve the optimal prediction, they have adopted this principle in their experiments. Additionally, the random splitting may not be suitable in case of time series-based datasets, where the item values have an impact on its next sequenced values. The other drawback is represented in the size of each split, which was two weeks to predict up to one week. One-week prediction could be considered a long-term prediction, where the far future is uncertain (suffers from accumulation of errors) in case of the same size of the training dataset [60, 61].

Authors, in [50], have proposed a method that conjunct the normal neural network and linear regression with the sliding window mechanism, where a window of a specific set of samples moves over the data (sample by sample) to compute the output for each input sample statistically [62], to predict CPU workloads. The authors have considered the analysis process of these workloads as a time series problem. One of the main drawbacks of this study is the usage of artificial datasets[1] instead of real ones for training and verifying the prediction model. Also, the size of these datasets is relatively small. Therefore, the obtained prediction model may not be suitable for real cloud data centers. Although the experimental results have shown that the sliding window mechanism has had the most impact on the accuracy of the proposed method, the authors have not clearly specified how the prediction model has been constructed with the use of this mechanism.

To address real-time and accuracy issues related to the usage of prediction methods for the management process of workload balancing, authors, in [20], have proposed a 2D parallel improved LSTM neural network method combined with an error backpropagation method for forecasting workloads. The authors have considered the analysis process of these workloads as a two-dimensional time series problem (i.e., day and time). Additionally, a parallelization mechanism has been considered as an attempt to achieve a real-time resource workload management process. Although the experimental results have shown higher accuracy and real-time performance, the authors have not clearly specified whether the proposed method is appropriate to predict all types of workloads or to a specific type, such as CPU, Memory, and Network.

In an attempt to reduce the energy consumption of cloud data centers through managing the workload balancing process effectively, authors, in [63], have proposed a method that depends on the linear regression and wavelet neural network mechanisms to provide short-term workload prediction considering data seasonality. One of the drawbacks of this method is using a simulated experimental environment (i.e., CloudSim) instead of a real one for the performance evaluation process. Although this method provides good performance, according to the authors' evaluation results, it is appropriate only to low workload data centers.

In order to address cloud technology challenges, such as dynamic scaling of resources and power consumption, authors in [8] have proposed a long short-term memory (LSTM) neural network-based cloud datacenters' workload prediction model. According to this study, the authors have used three benchmark datasets of web server logs. Using a non-recent enough and non-cloud related dataset (i.e., data values in the dataset were collected by monitoring non-cloud resources) represents, from our point of view, the main drawback of this study. The used dataset may not be suitable for the evaluation process due the dynamic and heterogeneity of the cloud environment.

In the context of cloud technology, authors in [19] have presented an experimental comparative study to identify the pros and cons of three workload prediction mechanisms; Autoregressive integrated moving average (ARIMA), Multi-Layer Perceptron (MLP), and Gated Recurrent Unit (GRU). The experimental results have shown, for short-time intervals' prediction, the three mechanisms achieved good results. Neural network mechanisms (i.e., MLP and GRU) have provided accurate prediction using fewer samples than ARIMA. Time-consuming due to the periodical update of the prediction model for each prediction time represents the main drawback of the ARIMA mechanism. Also, the evaluation procedure may suffer from some blurring, where the source or technique of obtaining the used dataset has not been mentioned clearly.

[1] Artificial workloads obtained by TPC-W workload generator

For improving accuracy prediction of cloud resources' workloads, authors in [35] have proposed a method that integrates the long short-term memory (LSTM) encoder-decoder neural network with the attention mechanism. The performance and effectivity of this method have been evaluated using two workload datasets collected from real clouds. The main drawback of this study is the small size of the used datasets (i.e., workloads for almost a period of eight days only). Additionally, the accuracy of long-term predictions has not been verified in this study.

For addressing challenges related to workload prediction in cloud environments, authors in [5] have proposed a prediction algorithm for workloads (L-PAW). The proposed algorithm can be considered as the integration of the top-sparse auto-encoder (TSA)[2] and Gated Recurrent Unit (GRU) recurrent neural network (RNN) mechanism. In order to evaluate this algorithm, real-world cloud workload datasets (i.e., from Google and Alibaba) have been used. The GRU limitation of using the same length for both input and output sequences may represent one drawbacks of the proposed algorithm. This drawback can be solved in my study by using a sequence-to-sequence algorithm that can provide more dynamism by permitting using various lengths for input and output sequences. Also, splitting the used datasets randomly into training and testing datasets may contradict the concept of considering the dependency of the workloads datasets.

Despite the great benefits of cloud technology, it suffers from some issues. One of these issues is the dynamic resource scaling and power wasting. Workload prediction models contribute greatly to addressing these issues effectively. This formed the main motivation for the authors in [18] to propose a workload prediction model using neural network and adaptive differential evolution algorithm. The performance of the proposed model has been evaluated using benchmark datasets (i.e., NASA HTTP traces and Saskatchewan HTTP traces), which may not accurately represent the workload nature in the cloud environments.

2.2 Statistical and Machine Learning Studies

In this subsection, the most related studies to statistical or machine learning based workload prediction of cloud resources are presented in the following paragraphs.

Authors, in [4], have proposed an adaptive forecasting model for workload and other cloud data center management processes. According to this study, six forecasting methods have been considered (i.e., Simple Exponential Smoothing (SES), Holt's Linear Trend (HOLT), Holt's Damped Trend (DHOLT), Auto-Regressive Integrated Moving Average (ARIMA), Linear Regression with Trend (LR), and TBATS) to predict the current state of resources in a cloud data center. Also, the authors have considered 77 data windows of different sizes (from 8 to 66 measurements) to specify the best combination of these methods and data windows. The dataset used to construct and evaluate the proposed models was a real-world dataset collected over one month, where the measurement interval is 5 min. From my point of view, restricting the predicted values to only one value (i.e., one-term prediction) and the probability of obtaining values outside the range [0% - 100%] represents the main drawbacks of this study.

Authors, in [49], have proposed a linear regression-based method for providing a short-term prediction of the CPU resources' consumption. This method depended on the last 12 reading values, which represents consumption over one hour ago (the reading interval of the used dataset was five minutes), to decide whether CPU usage of a specific VM is over-loaded or not according to the current value (short-term prediction). Using a fixed number of readings (twelve) to build a prediction model may represent one of drawbacks of this method. In real environments, this may not be suitable for all scenarios of the CPU consumptions. Additionally, better accuracy can be obtained using other prediction methods, such as Artificial Neural Network (ANN) [64]. In real data, assuming that there is a linear relationship between variables is incorrect many times. So, linear regression is not recommended by most applications [65].

Authors, in [7], have proposed an energy-aware resource allocation framework in a cloud environment. This framework expects the number of VM requests and the amount of their associated CPU and memory usage, as well as, it expects number of the needed physical machines in order to reduce energy consumption. The prediction mechanism starts with categorizing workloads of a specific VM into classes and predicting the number of requests for a specific class using the Wiener filter. To evaluate the proposed framework, the

[2] *Sparse autoencoder* is an auto-encoder with linear activation function, where the *highest* activities in hidden layers only are kept.

authors used Google dataset collected over one month from a cluster of 12,500 Physical machines. According to this work, predictions should be calculated every minute to maintain the adaptability with the current state of VM. This may increase the power and computing consumption.

Authors in [13] have proposed two Markov-based workload prediction models for CPU cloud resources; Continuous Time Markov Chain (CTMC) and Discrete-Time Markov Chain (DTMC) models. Dataset used in this study is real-world data collected from a Google cluster of 11 k machines. Results have shown that the CTMC-based model provides performance and efficiency better than the DTMC-based model. Additionally, the authors have proved that the prediction-based mechanisms, such as CTMC, outperform others that do not employ prediction, such as the Grid Resource Information Retrieval (GRIR) mechanism, in maintaining workloads balanced in cloud resources.

Authors, in [21], have proposed a web applications' workload prediction model that is composed of three prediction mechanisms (i.e., linear regression, ARIMA, and support vector regression). For evaluating the proposed model, the authors have used real-time web application datasets. Applying this model for predicting workloads related to non-cloud applications, where instability and dynamic nature the dominant features, is the main drawback.

Accurate prediction of cloud resource workloads results in addressing some cloud computing challenges, such as auto-scaling and load-balancing. Therefore, authors in [10] have proposed a self directed workload forecasting method (SDWF) that depends on calculating the deviation in fresh forecasts for enhancing the accuracy of the future forecasts (i.e., forecasting error). The SDWF method utilizes an optimization algorithm called Blackhole to improve accuracy of the predicted workloads by organizing the learning dataset into a set of classes. The proposed method has been evaluated using real world datasets. Due to the instability and sequence of workloads, dividing data traces into two portions of fixed lengths (i.e., 60% for training data and 40% for testing data) might contradict the actual representation of the used datasets.

For the cloud computing environments, authors in [66] have proposed a workload prediction model called POSITING. This model is based on the sequential pattern mining for extracting workload patterns. Despite the accepted results of the POSITING model, it could not adapt to the instability and variation of workloads caused by the dynamical nature of these environments. For addressing drawbacks, the same authors in [14] have proposed an improved version of the POSITING model by exploiting the pros of online learning and the episode mining techniques. The proposed model has been evaluated using real and synthetic datasets. The main drawback of this work is the missing of precise information about episodes.

For improving the efficiency of workload predictors in cloud environments, authors in [67] have proposed a pattern-mining engine. Identifying and omitting redundant patterns from workloads represents the key task of this engine. Disregarding the dynamic change of workloads in the cloud environments represents one of the main shortcomings of this study.

Authors in [68] have proposed a Reinforcement Learning (RL) based technique combined with fuzzy to predict workloads of cloud resources. RL is a machine learning area concerned with learning values of actions in particular states. Unfortunately, RL-based techniques might lead to an overhead of states. Also, these techniques require a huge volume of data to build their prediction models. Additionally, their computational time is very high [69].

Authors in [70] have proposed an integrated workload prediction technique, which combines the Savitzky-Golay (SG) filter and Wavelet Decomposition (WD) with stochastic configuration networks. According to this technique, SG first smooths workloads, and the smoothed version of workloads is then decomposed into multiple components using WD. One of the main drawbacks of WD is to choose a suitable number of decomposition levels for the target dataset. Also, it suffers from shift variance that may change the actual concept of the dataset [71].

The main aim of cloud resource management schemes is to smoothing cloud issues, such as minimizing power consumption, and avoiding resource wastage, at low cost. The workload prediction plays an important role in improving these schemes. It provides early estimations of future demands. These estimations help clouds in assigning necessary resources to new or existing applications. In the same context, authors in [72] have studied the performance of several nature-inspired based metaheuristic techniques for predicting workload in cloud environments.

The main drawbacks of the covered studies can be summarized in the following points:

1 Statistic-based workload prediction studies assume the stationary of workloads that contradicts dynamic nature of cloud resources [4].
2 Machine-learning based prediction studies assume the independence of workloads. This assumption contradicts the correlated nature of cloud workloads [35].
3 Although RNN deep learning based studies consider the dependency of workloads, they fail dealing with long-term predictions [5].
4 The covered workload prediction studies in this paper adopt the cross validation procedure for building and evaluating their prediction models. This procedure ignores important properties related to workloads, such as instability and dependency. In addition, the resampling process of datasets followed by that procedure renders the consumed time for building the models is very long [42, 43].
5 In spite of the importance of the memory resource, most of the covered studies are only care about the CPU resources. In this paper, both resources are considered.

3 Preliminaries

An overview of Long-Short Term Memory LSTM) and Sequence-to-Sequence neural network models is presented in this section.

3.1 Long Short Term Memory (LSTM)

RNN models are appropriate to process time series data (i.e., sequence) because these models take into account relationships between the former states and the latter states [35]. On the other hand, all models generated from multilayer neural networks using the backpropagation algorithm (e.g., RNN models) fail to provide accurate results for long sequences (i.e., long dependency problem) due to vanishing gradient problem [73]. The greater number of time steps (i.e. memories), there is a greater chance of the back-propagation gradients that vanish down to nothing.

As an effective solution for the long dependency problem, gated neural networks, such as LSTM networks that are subset of RNNs, were presented. Due to the great results of the LSTM networks on various problems, they have become an alternative and popular choice than the basic RNNs. So, often, whenever an RNN is quoted, it usually refers to LSTM network [47]. This network delivers promising results regarding many long sequence-learning tasks [74]. Therefore, the LSTM models are suitable for addressing the problem of long sentence translation [46].

The LSTM networks consist of a set of LSTM cells. As shown Fig. 2, the structure of these cells consists of a hidden state ($\boldsymbol{h}$), cell state ($\boldsymbol{c}$), and three gates (i.e., Input gate "$\boldsymbol{i}$" , Forget gate "$\boldsymbol{f}$" , and Output gate "$\boldsymbol{o}$" that decides whether to update the cell state with the input, forget the memory from the last step, and output the memory, respectively). At time step $\boldsymbol{t}$, given the input $\boldsymbol{x_t}$, the current hidden state ($\boldsymbol{h_t}$) and the cell state ($\boldsymbol{c_t}$) are calculated as follows [35]:

$$f_t = \sigma\left(w_f.[h_{t-1}, x_t] + b_f\right) \tag{1}$$

$$i_t = \sigma(w_i.[h_{t-1}, x_t] + b_i) \tag{2}$$

$$\widetilde{c}_t = \tanh(w_c.[h_{t-1}, x_t] + b_c) \tag{3}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \widetilde{c}_t \tag{4}$$

$$o_t = \sigma(w_o.[h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t \odot \tanh(c_t) \tag{6}$$

Where $\boldsymbol{f_t}$ denotes the forget gate output at time step $\boldsymbol{t}$, $\boldsymbol{i_t}$ denotes the input gate output at time $\boldsymbol{t}$, $\widetilde{c}_t$denotes candidate values to be added to $\boldsymbol{i_t}$ at time $\boldsymbol{t}$, $\boldsymbol{o_t}$ denotes the output gate output at time $\boldsymbol{t}$, $\boldsymbol{w_i}$ and $\boldsymbol{b_i}$ denote the weight matrixes and bias for the input gate layer, $\boldsymbol{w_c}$ and $\boldsymbol{b_c}$ denote the weight matrixes and bias for the candidate values' computational layer (i.e., cell state), $\boldsymbol{w_f}$ and $\boldsymbol{b_f}$ denote the weight matrixes and bias for the forget state, $\boldsymbol{w_o}$ and $\boldsymbol{b_o}$ denote the weight matrixes and bias for the output gate at time step $\boldsymbol{t}$, $\boldsymbol{\sigma}$ is the sigmoid function, $\boldsymbol{tanh}$ is the hyperbolic tangent function, and $\odot$expresses element-wise multiplication.

3.2 Sequence-to-Sequence (Seq2Seq)

Seq2seq is an encoder-decoder framework for translating one sequence (i.e., string) into another using RNN, where the gated neural network (i.e., LSTM) is used for avoiding vanishing gradient problem [46]. Seq2seq models are widely used with promising results for addressing challenges related to many areas, such as language translation, dialog systems, image captioning, and text summarization. Unlike the normal neural network, preserving the order of the inputs, where the context for each element is the output from the previous step, is the key thing that describes the seq2seq models. So, these models allow processing information, which the time or the order of time should be considered. Also, they encode only the necessary information [47].

As shown in Fig. 3, the simple structure of the seq2seq model consists of two primary and separate components (i.e., encoder RNN and decoder RNN). At each time step (***t***), the encoder takes one item as an input ($\boldsymbol{x_t}$). Through a set of time steps (***n*** steps or the length of input sequence), the entire sequence of items is turned into a context vector (***A***) or so-called the encoder's hidden states. The decoder takes that vector to be turned into the desired output sequence (i.e., $v_1 \ldots v_k$) of length ***k*** items, where the output word (v_i) of each decoder step along with the decoder hidden states (B) from this step are considered as an input to the next step [47].

3.3 Sequence-to-Sequence with Attention

Traditional seq2seq models fail to capture the essence of the entire sentence (i.e. sequence of workloads) and goes worse as the sequence length increases [47]. During the decoding process, these models assign the same weights for the historical workloads, although their effect on the current workload is various. Assigning relevancy scores, using what is so-called *Attention score* mechanism, to the preceding workloads for evaluating their differentiated importance may be better for addressing such situations [35]. Therefore, combining the attention mechanism with the seq2seq technique may help assign expressive weights for the corresponding decoding time steps [35].

Compared to the basic seq2seq model, combing the attention mechanism achieves better efficiency on the short-term sequences. The attention mechanism enables remembering 50 time steps instead of only 30 in case of the basic models [47]. Additionally, this mechanism relieves the encoder from the necessity to encode all information in the input sequence into a fixed-length vector (i.e., context vector of the last encoder time step). Instead, the essence of the input sequence will be spread throughout all the encoder time steps. Therefore, the context vector of every encoder time step will be considered in the decoding process. The generated vectors can be selectively retrieved by the decoder accordingly [75]. As shown in Fig. 4, the attention-based seq2seq modeling considers capturing information from the entire input sentence. Hence, the decoding process depends on the combined weights of all the encoder states (i.e., Real Context) instead of only the last encoder time step [47].

3.4 Continues Time Markov Chains Based Time-Series Prediction Model

The field of artificial intelligence has adopted stochastic techniques for machine learning operations. Bayes' theorem is the basis for these techniques where Bayesian methods can predict future events from historically recorded events. Markov Chain Model (MCM) is one of these methods. In MCM, the probability value at a specific time or a given location along a sequence is a function of the probability values taken at prior time periods or locations [13].

In MCM, the number of historical locations (i.e., states) affect the occurrence probability of a state at a specific time ***i***. In the first-order MCM used for the comparative evaluation process in this paper, this probability is only dependent upon the state located at location or time (***i-1***) [44]. The first-order MC model is defended by $K + K^2$ probabilities, where K is the number of states. The K probabilities or so-called initial probabilities (π) captures the occurrence probability of the **K** states at the first position in the given sequences, while K^2 captures the conditional probabilities of observing a state $\boldsymbol{K_x}$ at position ***i*** given the state $\boldsymbol{K_y}$ at position (***i-1***). These conditional probabilities are expressed in so-called $K \times K$ transition matrix [13].

According to the Continuous Time Markov Chain (CTMC) [13], the state transitions happen at any period of time. CTMCs move from one state to another according to a transition matrix as in MCM combined with a set of time rates. The time spent in each state is exponentially distributed with parameter λ, where every time state ***i*** is visited, the chain spends there, on average $1/\lambda_i$ time units before moving on.

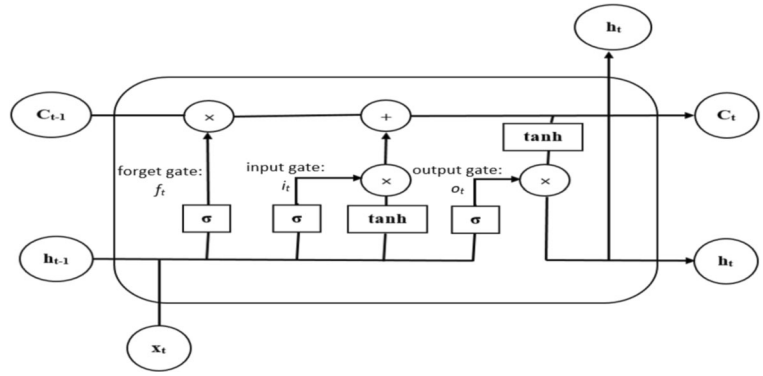

Fig. 2 Structure of the LSTM cell [35]

4 The Proposed Workload Prediction Model

According to my conducted survey study, it is better to consider the workload problem as a time series prediction problem. During the research process among the common techniques to address such problems, it is found that the neural network deep learning techniques provide solutions outperforming the other machine learning prediction techniques and statistic-based methods. By digging deeper into the neural techniques, it is observed that techniques related to addressing translation and question-answering problems, such as encoder-decoder techniques, represent the most suitable. Therefore, in this paper, the workload prediction problem in cloud computing environments is treated as a translation problem. Hence, the Attention-Seq2Seq-based NMT technique is adopted for addressing this problem by predicting future states of cloud resources.

Additionally, a novel validation procedure (i.e., CumP) is proposed for validating the robustness of the constructed workload prediction model. So, the CumP is evaluated compared to the CrossP, which is the most common validation procedure, in terms of accuracy and computational time. The construction of the proposed model and the algorithmic steps are described in the following subsections.

4.1 Workload Prediction Model Preparation

The construction of the proposed model depends on the formulation of the workload problem as a translation problem. Suppose there are a set of workload readings (i.e., training dataset ***D***) collected from specific resources (e.g., CPU and Memory usage percentage), these readings can be reformulated as follows (see Fig. 5):

(1) As an initial step, the given workloads are clustered into a suitable number of classes (***K***) using a simple, fast, and common clustering algorithm, i.e., *K*-means, where other algorithms may be more expensive in cloud environments [13].
(2) The given data are quantized into the generated classes. In other words, an alphabet character that indicates its assigned class replaces each workload value.
(3) The quantized dataset should be grouped into blocks (i.e., words; w_1, w_2, w_3…etc.) of fixed or variable window length to form a set of adjacent words. In this paper, the word length is assumed to be fixed (e.g., ***W***). These words are formulated into (X, V) pairs; the formulated dataset (***FD***) = $\{(w_i, w_j) \mid w_j$ is the translation of w_i, $i = 1, 2, 3\ldots$, and $j = i + 1\}$.

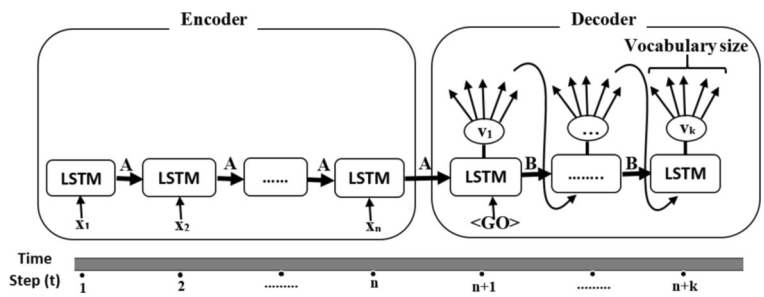

Fig. 3 A simple Seq2Seq structure

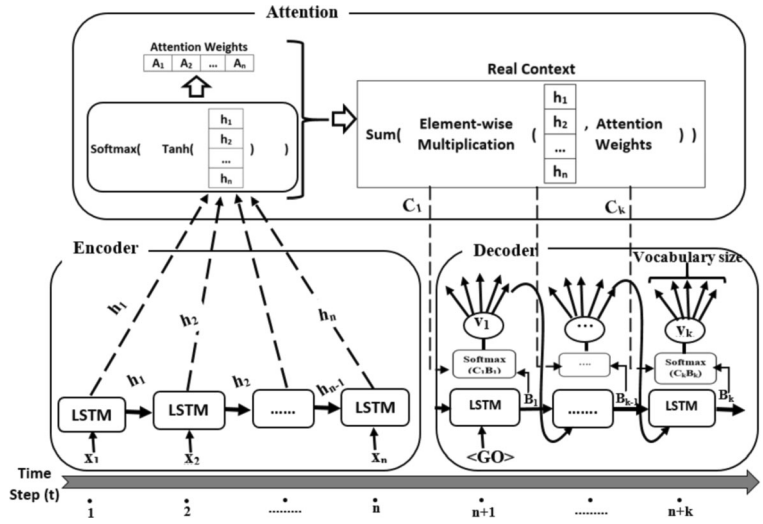

Fig. 4 A simple Attention-based seq2seq structure

4.2 Smoothing and Discretizing Data

Despite the effectiveness of using a clustering technique in the data formulation process, the error caused by noisy data could affect the next phases of the model construction. In this work, this effect is evaluated compared to use a discretization technique.

Data collected for constructing prediction models may have unwanted values (i.e., noise), which negatively affect the extraction process of actual relationships among values. Noisy data have a significant impact on the predicted values as they provide fake patterns and miss important patterns, which dramatically led to poor prediction results [76]. One of the simplest and most common techniques used for smoothing datasets is the Savitzky-Golay filter (SG) [49, 77]. Getting rid of high-frequency noise while preserving the original shape of data represent the key advantages of SG (all properties of the SG filter are presented in [78]). For each data point, the SG assigns a least square fit with a predefined-order polynomial (i.e., ***polyorder***) over an odd-length window (i.e., ***window_length***) centered at this point [79].

For easy and faster data processing, filtered continuous data should be transformed into categorical units by grouping them. One of the common and effective techniques used for discretizing data is the binning process (i.e., grouping data into "***bins***" of equal width). One of the popular libraries used for this process is the python library "pandas.cut" [80]. It is used interchangeably with the K-means clustering technique to measure the impact of the smoothing technique.

47	44.6	41.1	64.4	63.7	66.9	45	31.5	31.7	30.8
49.1	42	37	37.1	31	31.5	37	41.2	36.1	41.5
44	48.8	42.3	45.7	37.9	46	37.3	13	35	53.3
…	…	…	…	…	…	…	…	…	…
⇓ K-means clustering & Savitz-ky-Golay filter									
C	I	I	H	H	H	I	A	A	A
C	I	E	E	A	A	E	I	E	I
I	C	I	C	E	C	E	G	E	C
…	…	…	…	…	…	…	…	…	…

⇓ Translation Pairs

(CIIHHHIAAA, CIEEAAEIEI), (CIEEAAEIEI, ICICECEGEC) …

Fig. 5 An example of formulating the workload problem as a translation problem using K-means, where $K = 9$ and $W = 10$

4.3 Algorithmic Steps of the Proposed Model Construction

Every FD pair encompasses of two items (X, V), where X is the input sequence to the encoder steps and V is the output sequence from the decoder steps (i.e., V is the translation of X). According to Fig. 4 discussed in Section 3.3, for every pair of the FD dataset, X (e.g., CIIHHHIAAA) is passed through the encoder steps (i.e., one step for each character/workload "x_i" in X). LSTM at last step should have the option of retrieving the hidden state for every workload "x_i" to be true (i.e., *return_sequences* = *true*) in order to achieve the full use of the input sequence X. This is the trick of the attention code to memorize long sequences.

For every decoder step, deciding, which of the retrieved hidden states is most associated to the current step, can be taken by calculating a weighted sum (i.e., Attention Weights) for these hidden states and feeding it as an input to the decoder stage. This sum is obtained by applying two activation functions; *tanh* and *softmax*, respectively (see Fig. 4). During the decoding stage, each encoder-hidden state should be multiplied (i.e., element-wise multiplication) by its attention weight for drowning out hidden states with low weights. Because of this multiplication, a set of real context vectors (i.e., C_1, C_2….etc.) are generated, one (i.e., C_i) for each decoder time step $\boldsymbol{i}$.

LSTM cell in the decoder step $\boldsymbol{j}$ processes its inputs (i.e., output $\boldsymbol{v_{j-1}}$ and hidden sate $\boldsymbol{B_{j-1}}$ from the preceded cell $\boldsymbol{j-1}$) to generate a new hidden state $\boldsymbol{B_j}$, while output of the cell is discarded. Inputs to the first cell are initiated randomly. For the output of the step $\boldsymbol{j}$, it is calculated as follows:

- Concatenating $\boldsymbol{C_j}$ and $\boldsymbol{B_j}$ into one vector $\boldsymbol{CB_j}$.
- Passing $\boldsymbol{CB_j}$ through a feedforward NN to generate the output $\boldsymbol{v_j}$.

Outputs of the feedforward NNs indicate the translated sequence V (i.e., CIEEAAEIEI).

4.4 The Proposed Cumulative-Validation Procedure

Tuning parameters effectively, in proportion to the nature of the collected datasets, play an important role in producing a robust prediction model with high-efficiency and performance. The common procedure for constructing prediction models depends on splitting the dataset into training dataset and testing dataset. For an accurate emulation of the real world prediction, we have to obtain a well understanding of the present to obtain an accurate prediction for the future [43]. When dealing with time series datasets, you have to be careful because they are collections of chronological data values that are so related to their historical ones. Therefore, in this paper, a cumulative-validation procedure (i.e., CumP) is proposed.

For the CumP, the given dataset is divided into $\boldsymbol{n}$ periods $\mathbf{P_i}$ of same length; $\mathbf{FD} = \{P_1, P_2, P_3, P_4 \ldots P_{n-1}, P_n\}$. These periods are accumulated to construct the model $\mathbf{M_{j-r}}$ for predicting the period $\mathbf{P_j}$, where $\{P_{j-r} \ldots P_{j-2}, P_{j-1}\}$ are accumulated with the same order and $\boldsymbol{r}$ is a constant that should be learned ($\boldsymbol{r}$ is set to one as a default value). Applying this procedure (see Fig. 6) results in $(\boldsymbol{n} - \boldsymbol{r})$ prediction models (i.e., M_1, $M_2 \ldots M_{n-r}$) that should be generated. The suitable model $\boldsymbol{j}$ for predicting workloads at specific time (e.g., $\boldsymbol{t} = 6500$ min since the launch of the application) can be specified according to the Eq. 7:

$$j = \left\lfloor \frac{t \times n}{T} \right\rfloor - r + 1 \tag{7}$$

where $\boldsymbol{T}$ is the total time length of the $\boldsymbol{n}$ periods entirely (i.e., week, month, or year in minutes or seconds) and $\boldsymbol{t}$ is greater than length (P) of M_1. For example, suppose that $\boldsymbol{T}$ of the collected data is 28 days (i.e., $\boldsymbol{T}$ = 28 × 24 h × 60 min = 8064 min) and these data are divided into 4 periods (i.e., $\boldsymbol{n}$ = 4), the suitable model for predicting values related to the fourth time-period (e.g., workloads at time $\boldsymbol{t}$ = 6500) should be the third model ($\boldsymbol{j}$ = 6500*4/8064 ≈ 3).

Considering n is the largest possible number of periods (i.e., length of P_i is the shortest possible length) that can provide an acceptable length of P_i to be used for learning effectively. This number varies according to the volume of the collected dataset. Each set of adjacent periods are accumulated to generate a suitable model for predicting their next adjacent period. The smallest effective number of periods to be accumulated should be specified. This number is assigned to the constant $\boldsymbol{r}$. In this paper, model $\mathbf{M_i}$ can be generated by accumulating one more adjacent period to the model M_{i-1}. Each time, the accuracy is measured. This accuracy is expected to be improved till

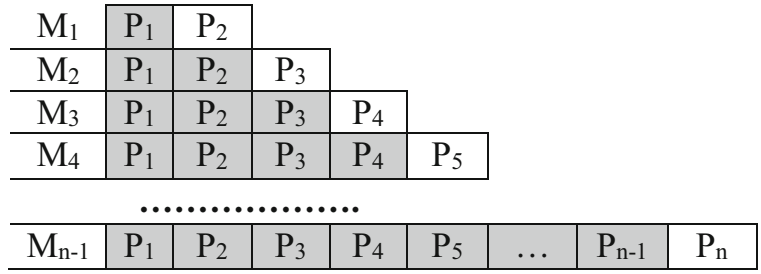

Fig. 6 An example of CumP. The data are split into ***n-1*** cumulative time-period datasets, where the training datasets are colored with the gray while testing datasets with white

specific number of periods, after that the accuracy will start to be degraded. This number represents ***r***.

For the CrossP procedure, the given dataset is split into ***n*** time-periods **FD** = {P_1, P_2, P_3, P_4…P_{n-1}, P_n}, where every time-period P_i is used as a testing dataset while the remaining time-periods are treated as training datasets. As shown in Fig. 7, applying this procedure produces ***n*** prediction models. The validation accuracy of these models is calculated as the average of their accuracies.

The algorithmic steps of the proposed Attention-Seq2Seq-based workload prediction technique can be summed up as follows:

(1) Preparing the collected workloads by specifying ***K*** (i.e., number of classes or bins) that can be learned, ***W*** (i.e., workload sequence length) that also can be earned, and the clustering algorithm (e.g., *K*-means) or the discretization algorithm (e.g., binning technique) to generate quantized formulated dataset (***FD***).
(2) Specifying the suitable learning procedure (i.e., CrossP or CumP) as well as ***n*** (i.e, number of time periods) to generate the needed workload perdition models using the Attention-Seq2Seq- based technique.
(3) During the prediction time, the suitable model is determined according to the Eq. 7 as discussed before.

5 Performance Evaluation

The implementation of the proposed technique is introduced in the following subsections.

5.1 Dataset and Implementation Environment

To evaluate the proposed technique in accordance with the proposed learning procedures, the considered dataset was released by Google in 2011. This data set represents twenty-nine days of the status information about a cluster of eleven-kilo physical machines operated as a single unit. It consists of a realistic mix of workloads, as this cluster comprises non-homogeneous machines. The platforms in that cluster can be classified into three different categories and a variety of memory/compute ratios (for more details see [81]). These categories include 126, 10 K, and 795 non-homogeneous machines for the first category (i.e., A), the second category (i.e., B), and the third category (i.e., **C**) respectively. Measurements (e.g., exact machine configurations, and exact numbers of CPU cores and bytes of memory) in the Google dataset are normalized into the configuration of the largest machines. Machines related to the category **C** have the top configuration [81]. Number of machines encompassed in this category is large enough (i.s., not small as in A or huge as in B) to be used for evaluating the proposed technique and procedure. Therefore, for simplicity and easy understand, workloads (i.e., CPU

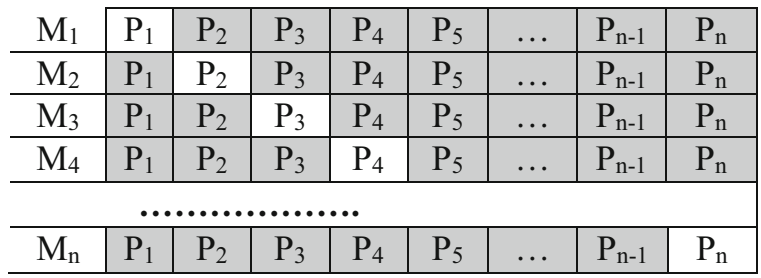

Fig. 7 An example of CrossP. The data are split into ***n*** time-period datasets, where the training datasets are colored with the gray while testing datasets with white

and Memory measurements) of this category's machines are only considered in this paper.

The used dataset includes usage percentages (i.e., workloads) of resources (e.g., CPU and Memory) by each task implemented on the Google cluster, as well as, requests to allocate these resources. In this paper, both CPU and memory usage percentages are focused. All of these workloads are stored in a table called "resource usage" that contains a measurement record every 300 s for 29 days for each task [13]. The experiments, in this paper, are conducted on the total CPU and memory usage of every machine, so the average CPU and memory usages of all tasks on every machine at every timestamp were added together to get these total CPU and memory values (i.e., workloads).

As a result, 795 files were produced. Each file has almost 8350 records of CPU and memory workloads and their timestamps. These workloads are divided into eight time-periods (i.e., n = 8 and period length is about one thousand workloads) as an initial division that will be validated with experiments. These periods are used to train and evaluate the proposed Attention- Seq2Seq-based technique as discussed before according to CrossP and CumP validation procedures.

As an example of workloads recorded in the produced data files, CPU and Memory usage percentages of two machines with ID "4,802,487,721" and "4,820,074,350", which were selected randomly from the category C, are shown in Figs. 8 and 9. It is observed that the Memory usage of the two machines is more stable than that of the CPU. This is due to killing tasks when the memory request exceeded its limit. But, this constraint is omitted in case of CPU workload, whereby tasks can use much more CPU than that of the requested CPU [82].

5.2 Implementation Environment

The experimental environment, in this paper, is a Laptop platform with Intel(R) Core(TM) i7-2620M CPU 2.70GHz and 8.0 GB memory. This platform has Windows 10 pro as an operating system. All programs needed to evaluate the proposed workload prediction technique are written using Python 3.8 as an Integrated Development Environment (IDE) with the TensorFlow 2.3.0 open-source software machine-learning package.

Experiments conducted in this work are divided into three groups. The main purposes of the first group can be summarized into: **a)** specifying the most effective value of K (i.e., number of classes). **b)** Evaluating the effectivity of using k-means clustering for discretization combined with the Savitz-ky-Golay (SG). **c)** Evaluating the efficiency (i.e., prediction accuracy and complexity) of the proposed Attention-Seq2Seq-based NMT Workload Prediction technique (AS2S). Finally **d)** evaluating the proposed cumulative-validation procedure (CumP). In this group, two datasets with IDs 4,802,487,721, and 4,820,074,350 are considered randomly.

For the second group, to check the stability of the proposed AS2S technique in terms of accuracy, experiments are expanded by employing the AS2S on the top 20% of machines (i.e., 159 dataset files) sorted descendingly by file size and IDs; 4,802,086,681, 4,802,092,023...etc. According to results of this group, the most effective length of periods is specified (i.e., the most proper ***r*** and ***n*** in Eq. 7 are specified).

Experiments of the last group are conducted on datasets that have achieved the worst accuracy compared to others in order to evaluate how improvement is achieved due to adjusting parameters (i.e., r, n, epochs, and latent dimensionality) specified in the previous group.

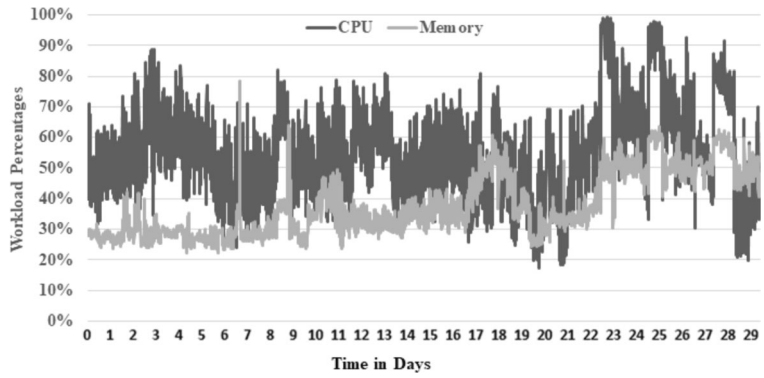

Fig. 8 Actual CPU and Memory workload measurements of the machine with ID "4,802,487,721"

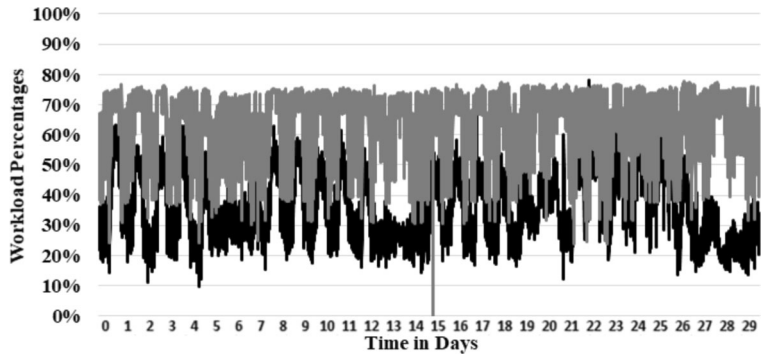

Fig. 9 Actual CPU and Memory measurements of the machine with ID "4,820,074,350"

5.3 Evaluation Metrics

To evaluate the performance of the proposed AS2S technique according to the proposed CumP validation procedure and the common one (i.e., CrossP), three metrics are used. The first metric is used to evaluate the accuracy of the generated CPU or Memory workloads. This metric is the Standard Deviation ($0 \leq SD \leq 1$) between the predicted and actual workloads. A high value of SD indicates a high prediction error and vice versa. When SD equals zero, the predicted workloads exactly matches the actual data and the prediction accuracy is 100%. This situation is impossible to occur as the predicted sequences consist of states represented by centroids. Of course, these centroids have a percentage deviation from their associated values.

In order to evaluate the accuracy of the generated workloads for both resources (i.e., CPU and Memory), a new metric called ***μSD*** is constructed. μSD is the average SD of CPU and Memory. This metric in turns can be used to calculate the global_SD, which is the average μSD of all machines.

The third metric is used to evaluate the computation time taken by the proposed technique to produce the required prediction model and to predict one value according to the adopted validation procedure.

5.4 Experimental Results

Results presented in this section are organized in a way that demonstrates the performance of the proposed AS2S technique, as a NMT technique, in terms of two parameters; prediction accuracy and complexity. Both parameters are determined according to the proposed CumP validation procedure and the common one (i.e., CrossP). These results are derived based only on the given datasets. Additionally, those results are obtained according to a specific set of parameters:

- Number of epochs (i.e., how many times should the training code be run on the training dataset entirely) is set to "**50**" as an initial value that will be tuned later.
- Batch size (i.e., number of samples fetched every step, where number of steps in one epoch equal the total number of samples divided by the Batch size) is set to **one.**
- Latent dimensionality (**Latent_dim**) of the encoding space (i.e., number of hidden states) is set to "**200"** as an initial value that will be tuned later**.**
- Activation function or so-called Transfer Function is set to "**softmax"**.
- Optimizer is set to "**rmsprop**" that is good, fast and the most popular optimizer used in deep learning [83]
- Error loss function or so-called cost function is set to "**categorical cross entropy"**.

These parameters are the most common ones for constructing Seq2Seq-based techniques. Values of those arguments are selected particularly as they have achieved the highest accuracy among a set of exploratory values. However, these values will be tuned in this paper later.

Experiments in this paper are organized into three groups presented in the following subsections.

5.4.1 Clustering and Validation Procedure Experiments Group

Experiments conducted in this group aim to specify the most effective value of K, evaluate K-means clustering

with SG filter as a discretization technique compared to the binning algorithm, and evaluate the proposed CumP procedure compared to the CrossP. In the same context, prediction quality and time complexity of the proposed AS2S technique is evaluated against two of the most common prediction time series techniques (i.e., CTMC as a representative of statistical-based techniques, and LSTM as a representative of deep learning-based techniques). These experiments are conducted under the same performance parameters' values and validation procedures.

For selecting the most appropriate value of K, the AS2S technique has been applied, under the same parameters' values, using various values for K (i.e., 3, 6, 9, 12, 20, and 40) on the workloads dataset of the machine "4,802,487,721". It should be noted that the effect of using the SG filter was taken away from this experiment. As shown in Fig. 10, the highest accuracy (i.e., the lowest SD for both CPU and Memory) has been achieved according to the given dataset at **K = 9**. It is also obvious that using small numbers of classes achieves accuracy higher than using large numbers. The fewer number of classes, the more frequent of these classes in the given dataset that in turns renders the higher the prediction accuracy. On the other hand, using fewer numbers than a specific threshold increases the deviation of the predicted workloads from the actual measurements. From my point of view, using a greater number of classes might reduce the deviation degree and thus increase the workload prediction accuracy, on the condition of growing the volume of the training dataset.

For evaluating the performance of the proposed AS2S technique in terms of accuracy, an experimental comparison has been conducted among this technique, the CTMC-based technique proposed in [13], and the LSTM-based technique. The three techniques have been applied on the same datasets (i.e., 4,802,487,721 and 4,820,074,350) using the same validation procedures (i.e., CumP and CrossP). As shown in Fig. 11 that demonstrates the global average accuracy of the three techniques over both datasets, the accuracy of the proposed AS2S outperforms the other two techniques. The AS2S has achieved accuracy about 94% (i.e., 1.0 – global_SD) using both validation procedures, while the CTMC has achieved accuracy about 89% using the same procedures. The LSTM has achieved 80% and 84% in case of the CrossP and the CumP procedures, respectively.

On the other hand, as shown in Fig. 12, the CTMC-based technique outperforms the proposed AS2S technique and the LSTM-based technique in terms of computational time. The CTMC has required about **2.6** microseconds using CumP and **5** microseconds using CrossP as the average computational time for constructing their required eight prediction models and performing the learning process, while the proposed AS2S has required about 346 s using CumP and 647 s using CrossP for constructing their eight models. The high computational time of the proposed technique was expected as the deep learning techniques (e.g., Neural networks-based techniques) do not care about learning time. They care only about accuracy. The more learning time, the higher the accuracy degree [84].

For the prediction time, as shown in Fig. 13, one word (i.e., sequence of workloads) of length 10 chars has required about 0.6 of second to be translated in case of the AS2S technique, 0.6 microseconds for the CTMC, and 0.014 of second for the LSTM.

For the effect of using the proposed CumP validation procedure on the performance of prediction techniques (e.g., AS2S and CTMC) in terms of accuracy, Fig. 11 shows that both the CumP and the CrossP have achieved

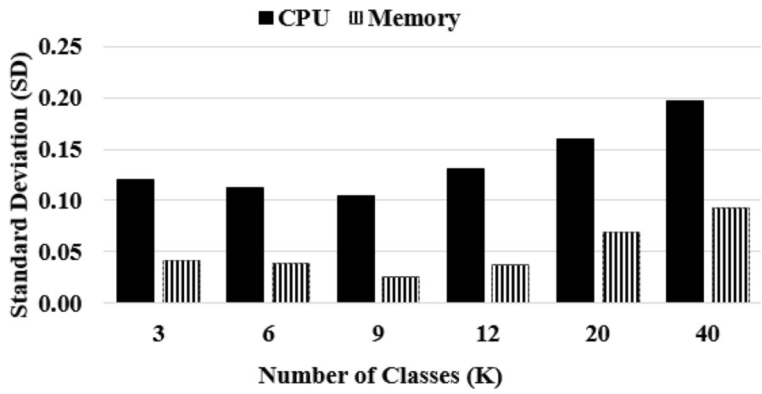

Fig. 10 Relation between number of classes and accuracy of the AS2S technique without SG filter given dataset "4,802,487,721"

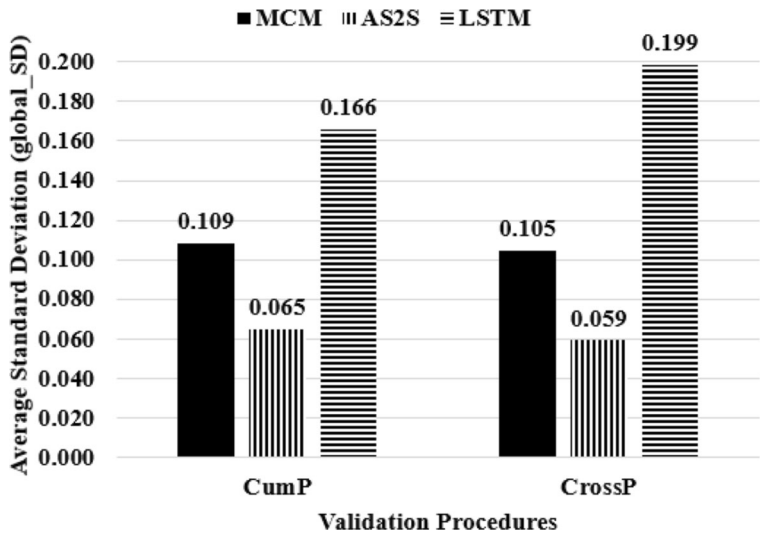

Fig. 11 The accuracy of CTMC-based, LSTM-based, and the proposed AS2S-based workload prediction techniques without SG filter using the proposed CumP and CrossP validation Procedures

a very close accuracy except in the case of the LSTM technique that have achieved better accuracy for the CumP. The difference in the recorded accuracy between the both procedures is not exceeding 0.006 (0.065–0.059) for the AS2S and 0.004 for the CTMC, but in the case of the LSTM, there is a relative superiority of the CumP over the CrossP by 0.033. On the other hand, as shown in Fig. 12, the CumP has had a significant effect on the computational time compared to the CrossP. The Cump has saved about 50% of time elapsed using the CrossP for the three techniques; 54% (0.0026 / 0.0048 ≈ 54%) for CTMC, 53% for AS2S, and 56.5 for LSTM.

For the effect of applying the SG filtering technique (default parameters; polyorder = 3, and window_length = 53) combined with the binning process (bins = K) for discretizing data (i.e., dataset "4,820,074,350"), instead of using the k-means clustering algorithm to formulate the collected data in the data preparation stage, Figs. 14 and 15 and Tables 1 and 2 show the importance of the SG filter. Both SG and binning together have achieved a superiority over the clustering alone by 4.3% (7.1% - 2.8% for the CumP and 6.8% - 2.5% for the CrossP) on the average of μSD metric. Filtering and discretizing improves the accuracy of the proposed AS2S-based workload prediction model to become about 97% on the average accuracy of the eight prediction models compared to about 93% for clustering alone. This underlines the importance of considering the filtering processes during the data preparation stage for getting rid of noisy data.

For using the binning discretization technique, picking the right range of each bin may represent a challenge. Also, it is not fair to associate the use of clustering techniques for data discretization with the negative impact on the accuracy without an experimental

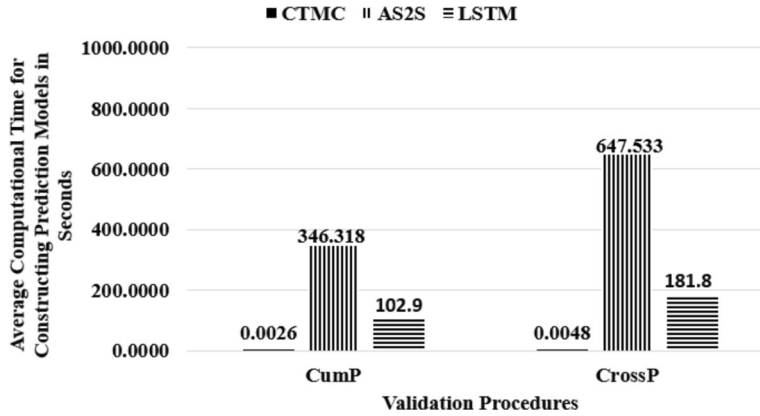

Fig. 12 The effect of the proposed CumP and the common CrossP validation procedures on the time elapsed by the proposed AS2S, the CTMC, and LSTM techniques for constructing the required prediction models

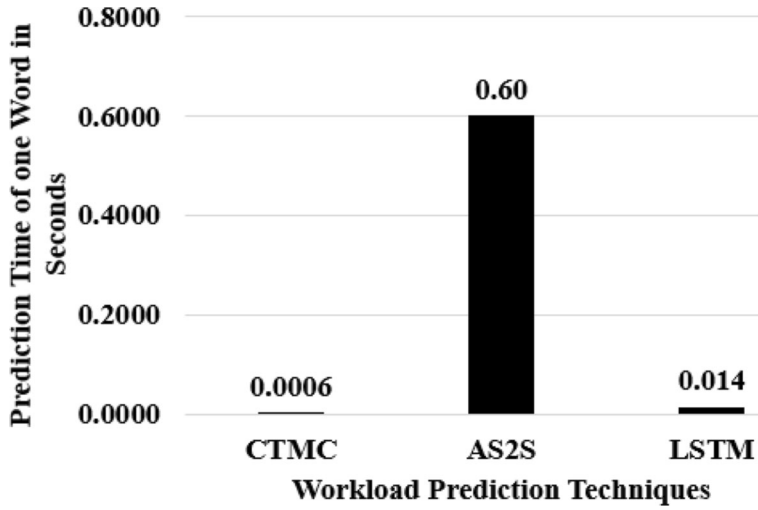

Fig. 13 Time elapsed by both the proposed workload pre-diction AS2S, the CTMC, and LSTM techniques to predict or translate only one word (sequence of workloads) of length ten workloads in seconds

verification. So, the experiment is reconducted with the same parameters, but the K-means clustering with the SG filter has been applied for data formulation. As shown in Fig. 16, clustering and filtering achieves the highest average accuracy (i.e., 98.1%) for the eight models (i.e., the lowest μSD 1.9%) followed by 97.2% (i.e., 2.8% on the average SD) for discretizing and filtering. This explains the great impact of the filtering techniques because the error caused by the noisy data could affect the next phases of the model construction. Also, the clustering techniques have a remarkable effect on the workload prediction accuracy.

All results indicate the efficiency of the proposed translation technique for predicting workloads in the cloud environments. It also indicates the role of the proposed cumulative validate procedure to improve the computational time needed to construct the prediction models by 50%, with an accuracy almost equivalence to that has been obtained using the cross-validation. Additionally, using clustering as data discretization techniques improves the accuracy of the prediction models.

5.4.2 Period Length and Average Accuracy Experiments Group

Experiments conducted in the second group have three aims: a) verifying the accuracy of the AS2S technique using more datasets. b) Defining the relationship between computational time and the number of training samples. c) According to the results of this group, the most proper r and n in Eq. 7 have been specified. For achieving these three aims, experiments of the first group have been expanded to include 20% of datasets under the same conditions (i.e., validation procedure = "CumP", r = 1, Epochs = 50, and Latent_dim = 200).

As shown in Figs. 17, 18, and 19, the results confirm the superiority of the proposed AS2S in terms of

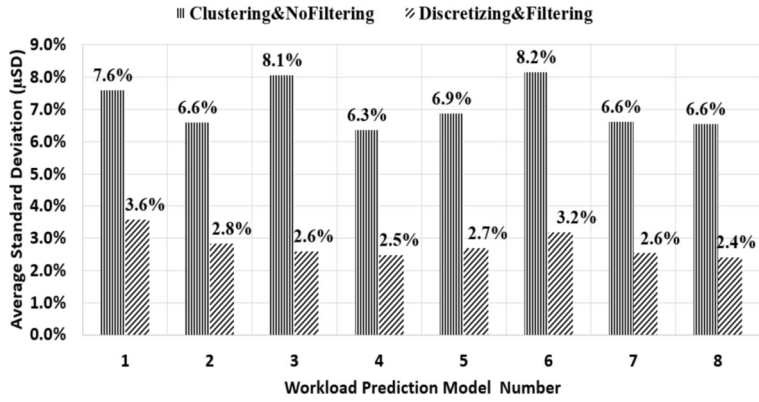

Fig. 14 Effect of considering filtering techniques followed by discretizing data of the machine with ID "4,820,074,350" compared to just using clustering for data formulation on the accuracy of the proposed AS2S-based workload prediction according to the CrossP validation procedure

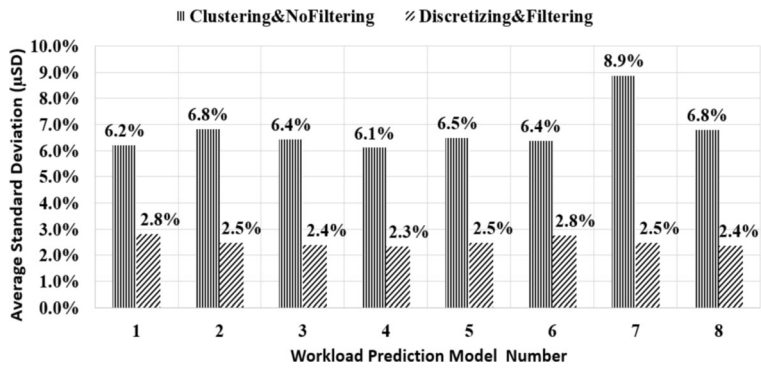

Fig. 15 Effect of considering filtering techniques followed by discretizing data of the machine with ID "4,820,074,350" compared to just using clustering for data formulation on the accuracy of the proposed AS2S-based workload prediction according to the CumP validation procedure

accuracy. It has achieved 96.8% on the average (96.4% for CPU and 97.2% for Memory) compared to 91% for CTMC and 86% for LSTM.

For the complexity of the proposed AS2S and CumP procedure, the time needed to construct one prediction model increases linearly by increasing number of training samples according to Eq. 8 (see Fig. 20).

$$\text{Time} = 0.0757\ \text{Training Samples} + 3.1571 \tag{8}$$

For space complexity, the proposed technique encompasses a set of variables for constructing each period-based prediction model. These variables can be summarized into: **a)** Encoder_Inputs that has the shape **($L\_FD \times W_e \times K_e$)**, and **b)** Decoder_Inputs that has the shape **($L\_FD \times W_d \times K_d$)**. "L_FD" denotes to number of word pairs "(X, V)" in the formulated dataset "FD", "W_e" to length of the longest word "X", "K_e" to number of characters in the alphabet of the "X" words, "W_d" to length of the longest word "V", and "K_d" to number of characters in the alphabet of the "V" words. For simplicity, W_e and W_d are assigned the same value "W". Also, K_e is assigned the value of "K", while K_d is set to be "K + 2" (e.g., K_d = 9 + start and stop characters = 11). For each character in the word "X", there are four functions (i.e., input gate, output gate, forget gate and candidate values) in the LSTM step. Each function has Latent_dim dimensionality (i.e., number of hidden states). In other words, there are 4 × Latent_dim for each step. **c)** In case of the encoder model, the input weighting matrix for predicting one character has the shape **($K_e \times$ 4Latent_dim)**, while the hidden state weighting matrix has the shape **(Latent_dim × 4Latent_dim)** and the bias vector has the size **4Latent_dim**. **d)** Things are very similar in the case of decoder model except K_d instead of K_e. **e)** In case of

Table 1 Accuracy of the AS2S-based workload prediction technique according to CumP and CrossP validation procedures with considering only clustering process to formulate data of the machine with ID "4,820,074,350"

Model Number	Clustering & No Filtering					
	CumP			CrossP		
	CPU_SD	Memory_SD	avg_SD	CPU_SD	Memory_SD	avg_SD
1	6.1%	9.1%	7.6%	4.9%	7.5%	6.2%
2	5.1%	8.1%	6.6%	5.0%	8.6%	6.8%
3	4.7%	11.4%	8.1%	5.1%	7.8%	6.4%
4	5.1%	7.6%	6.3%	4.7%	7.5%	6.1%
5	5.2%	8.5%	6.9%	5.2%	7.8%	6.5%
6	6.7%	9.6%	8.2%	5.0%	7.8%	6.4%
7	4.8%	8.5%	6.6%	8.7%	9.1%	8.9%
8	4.1%	9.0%	6.6%	5.5%	8.1%	6.8%
Average	5.2%	9.0%	7.1%	5.5%	8.0%	6.8%

Table 2 Accuracy of the AS2S-based workload prediction technique according to CumP and CrossP validation procedures with considering both filtering and discretizing data of the machine with ID "4,820,074,350"

Model Number	Discretizing & Filtering					
	CumP			CrossP		
	CPU_SD	Memory_SD	avg_SD	CPU_SD	Memory_SD	avg_SD
1	4.3%	2.9%	3.6%	3.1%	2.0%	2.8%
2	3.4%	2.3%	2.8%	3.1%	2.1%	2.5%
3	2.7%	2.5%	2.6%	3.3%	2.2%	2.4%
4	2.6%	2.3%	2.5%	2.8%	2.2%	2.3%
5	2.9%	2.5%	2.7%	3.1%	2.2%	2.5%
6	3.9%	2.5%	3.2%	2.7%	2.3%	2.8%
7	2.7%	2.4%	2.6%	4.3%	2.4%	2.5%
8	2.7%	2.1%	2.4%	2.8%	2.3%	2.4%
Average	3.2%	2.4%	2.8%	3.2%	2.2%	2.5%

Dense layer (i.e., final output layer), the weighting matrix has the shape **(Latent_dim × K_d)** and the bias vector of this layer has the size K_d. Therefore, the space complexity for each model can be denoted by $(2L\_FD \times W \times K + 8Latent\_dim^2) \approx O(L\_FD \times W \times K)$.

For studying the effect of the learning period length, experiments have been reconducted using various accumulative lengths $r \times 10^3$, where $r = 1, 2 \ldots 7$. As shown in Fig. 21, the average accuracy of the proposed AS2S technique for a wide range of datasets improves progressively by increasing the value of r up to a specific value (i.e., $r = 3$) then it begins to degrade. According to the experimental results, relation between the accuracy and the length of the learning period can be expressed polynomial in terms of μSD, r, n, and T using Eq. 9.

$$\mu SD = \frac{8\left(\frac{T \times r}{n}\right)^2 - 54\left(\frac{T \times r}{n}\right) + 372}{10^4} \quad (9)$$

5.4.3 Experiments Group for Tuning Parameters

The aims of the third group experiments can be summarized into two points: 1) tuning parameters of the

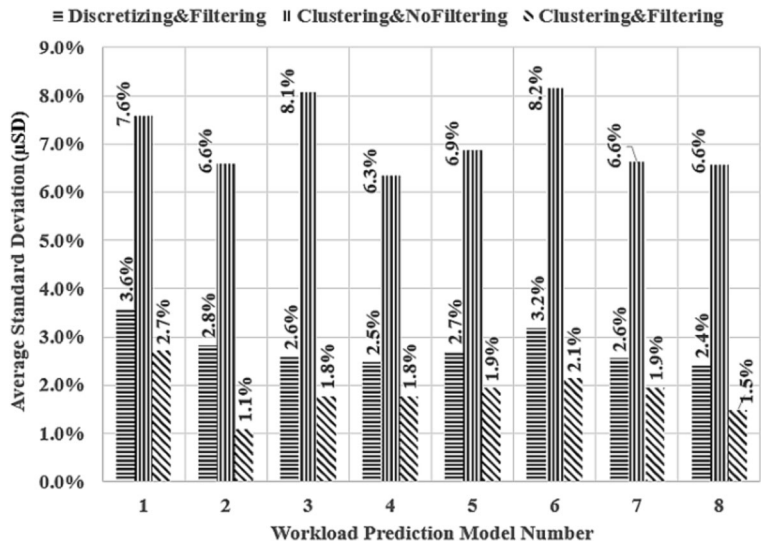

Fig. 16 Effect of considering filtering techniques followed by discretization interchangeably with clustering for formulating data of machine with ID "4,820,074,350" on accuracy of proposed AS2S-based workload prediction according to CumP validation procedure

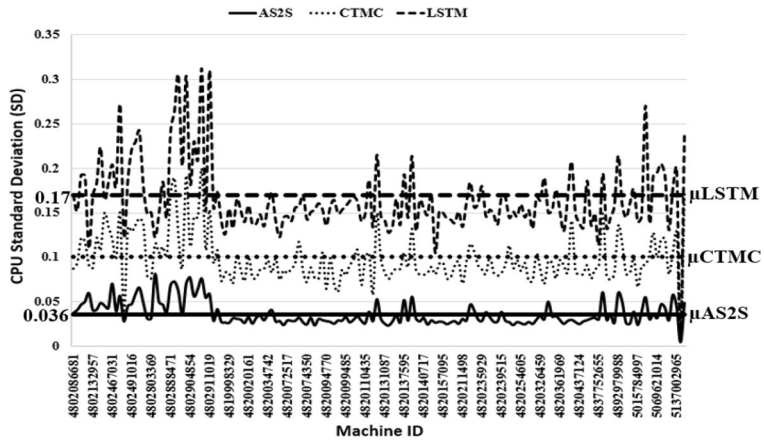

Fig. 17 The CPU accuracy of CTMC-based, LSTM-based, and the proposed AS2S-based workload prediction techniques for a wide range of datasets using the proposed CumP validation Procedure

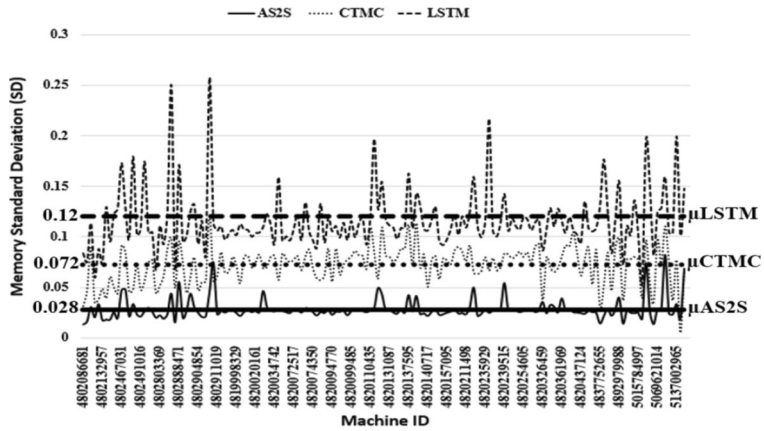

Fig. 18 The Memory accuracy of CTMC-based, LSTM-based, and the proposed AS2S-based workload prediction techniques for a wide range of datasets using the proposed CumP validation Procedure

proposed AS2S NMT technique for more improvement in time complexity and accuracy, and 2) Evaluating the improvement rate due to these tuned parameters.

For defining the proper value of the Epoch Parameter, the AS2S has been applied on a randomly selected dataset (i.e., 4,802,086,681) for various numbers of

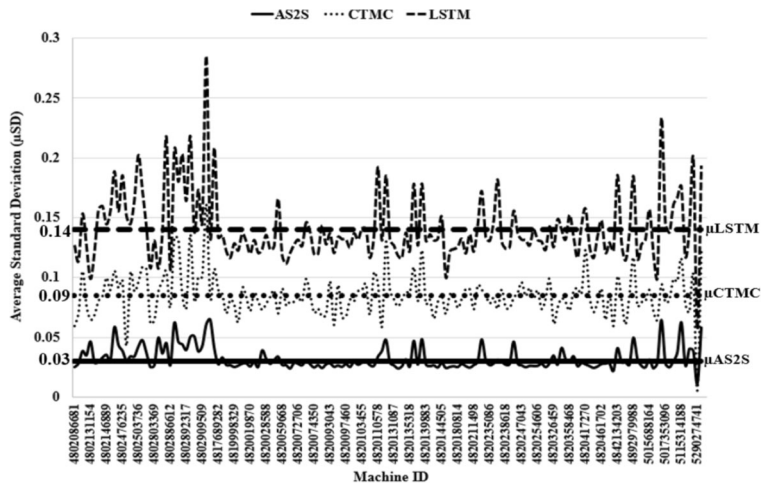

Fig. 19 The average accuracy of CTMC-based, LSTM-based, and the proposed AS2S-based workload prediction techniques for a wide range of datasets using the proposed CumP validation Procedure

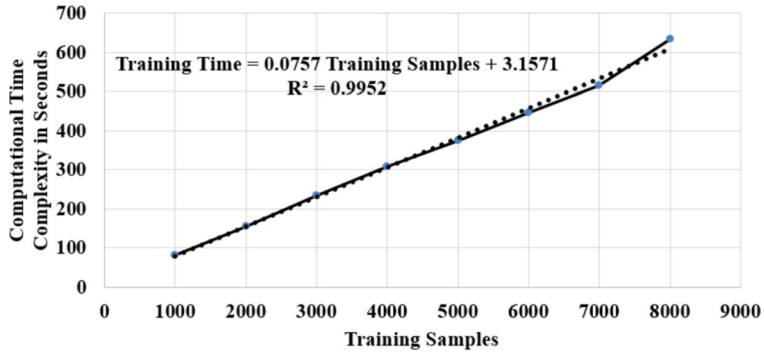

Fig. 20 Relation between the number of training samples and the computational time according to the proposed AS2S and CumP validation procedure

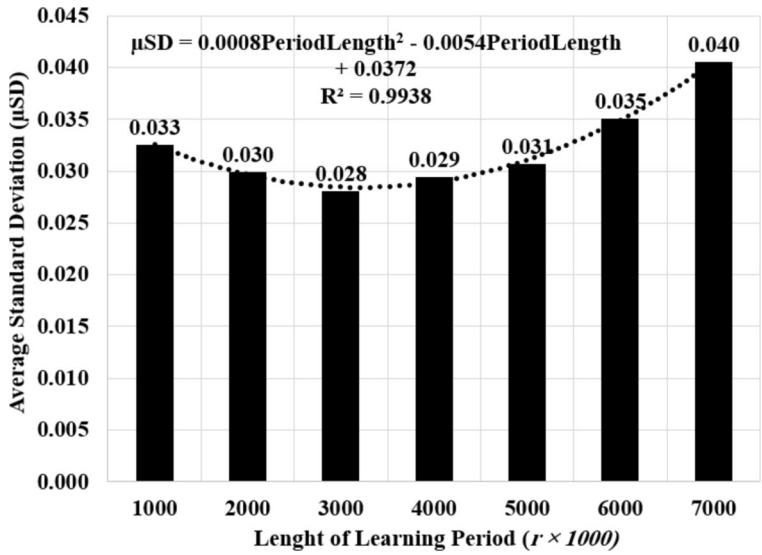

Fig. 21 The effect of the learning period length on the average accuracy of the prediction model over a wide range of datasets

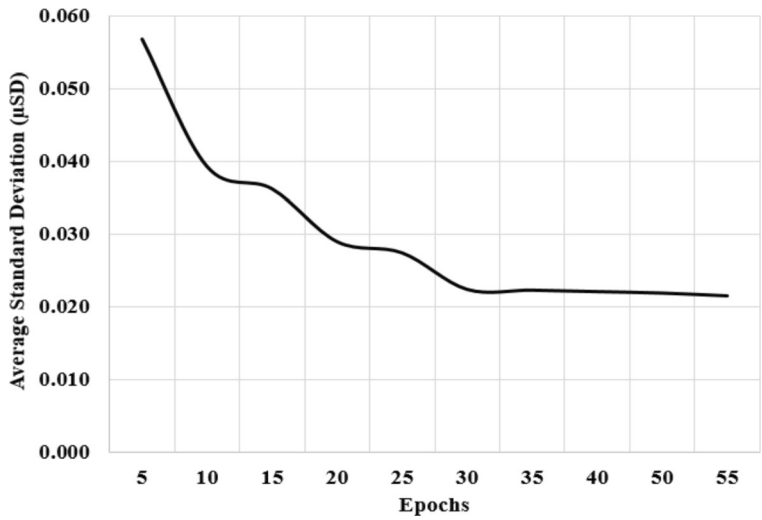

Fig. 22 The effect of using various numbers of epochs on the average accuracy of the proposed AS2S technique

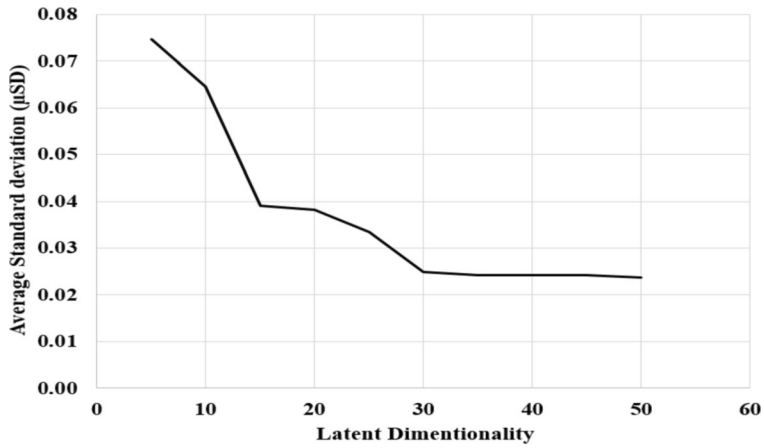

Fig. 23 The effect of using various values of Latent_dim on the average accuracy of the proposed AS2S technique

epochs. Results, in Fig. 22, show that the accuracy gets better by increasing the number of epochs up to specific value (i.e., ***epoch = 30***). After that, a very simple fluctuation, which can be neglected, is noticed.

For defining the proper value of the ***La-tent_dim*** Parameter, the AS2S has been applied on the same previous dataset for various values of hidden states or so-called latent dimensionality. Results, in Fig. 23, show that the accuracy gets better by using larger values of Latent_dim up to a specific value (i.e., Latent_dim = 30). After that, the accuracy has nearly recorded the same level. According to these results, the value "30" is selected as the proper value for each Epoch and Latent_dim parameter.

For evaluating these adjusted parameters (i.e., Epoch = 30, Latent_dim = 30, and r = 3), the AS2S has been applied again on the datasets that have recorded the worst µSD in the second group of experiments (i.e., Epoch = 50, Latent_dim = 200, and r = 1). As shown in Table 3, these adjusted parameters have improved the prediction accuracy by 1.1% on average. In other words, according to the proposed CumP validation procedure and the concept of dividing the prediction model into set period-based models, the proposed AS2S has improved

Table 3 The effect of the tuned parameters on the average accuracy of the AS2S using the CumP validation procedure

Dataset ID	µSD Before Tuning	µSD After Tuning	Improvement Accuracy rate
4,802,467,031	**0.071**	**0.046**	**2.5%**
4,802,888,471	**0.063**	**0.058**	**0.5%**
4,802,910,981	**0.080**	**0.060**	**2.0%**
5,017,329,882	**0.037**	**0.032**	**0.5%**
5,115,314,188	**0.073**	**0.061**	**1.3%**
5,777,397,608	**0.065**	**0.063**	**0.2%**
4,802,889,220	**0.050**	**0.034**	**1.6%**
4,802,517,408	**0.054**	**0.042**	**1.2%**
4,802,863,684	**0.058**	**0.044**	**1.4%**
4,802,904,615	**0.055**	**0.049**	**0.6%**
4,802,904,854	**0.059**	**0.032**	**2.6%**
4,820,115,846	**0.046**	**0.040**	**0.5%**
4,820,139,637	**0.047**	**0.040**	**0.7%**
4,820,222,428	**0.054**	**0.053**	**0.1%**
4,892,979,988	**0.046**	**0.039**	**0.7%**
Global_SD	**0.057**	**0.046**	**1.1%**

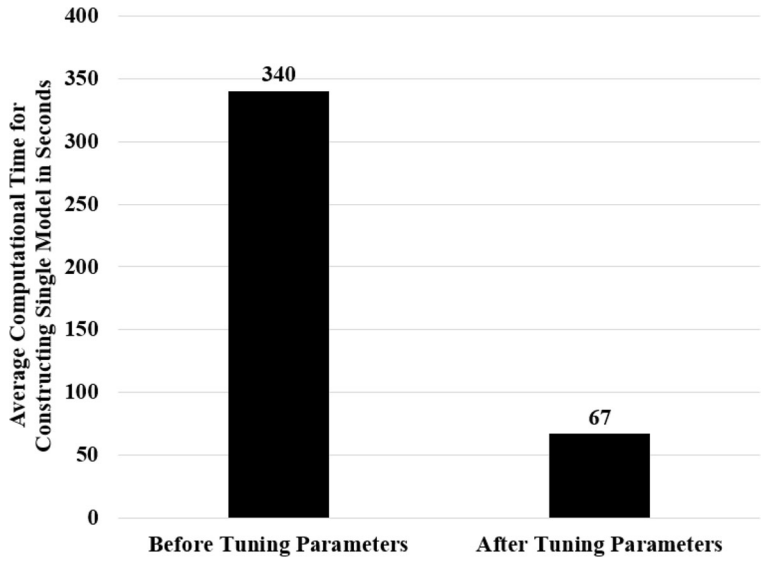

Fig. 24 The effect of tuning parameters of the proposed model and length of the learning period on the average computational time needed for constructing a single prediction model

the workload prediction accuracy to become 98.1% (97% + 1.1%).

Of course, these adjustments have had a clear effect over the computational time. As shown in Fig. 24, the average time taken for constructing a single period-based model is improved by 80.2%. The proposed AS2S has recorded 67 s on average compared to 340 s recorded before the adjustment process.

For defining the proper length of the predicted word (i.e., ***W***), the AS2S has been applied on a randomly selected dataset (i.e., 4,837,752,655) for various lengths. Results, in Fig. 25 demonstrates that the workload prediction accuracy is inversely proportional to ***W***, as shown in Eq. 10.

6 Conclusion and Future Work

As mentioned previously, the main objectives of this paper can be summarized into four points. The first objective is to study the effectiveness of using NMT techniques to predict the accuracy of workload time-series in cloud environments compared to other time-series prediction techniques, such as CTMC and LSTM. Results show that the proposed NMT-based AS2S technique provides a remarkable superiority in terms of accuracy by 98.1% compared to 91% and 85% for other sequence-based techniques, i.e. CTMC and LSTM models, respectively.

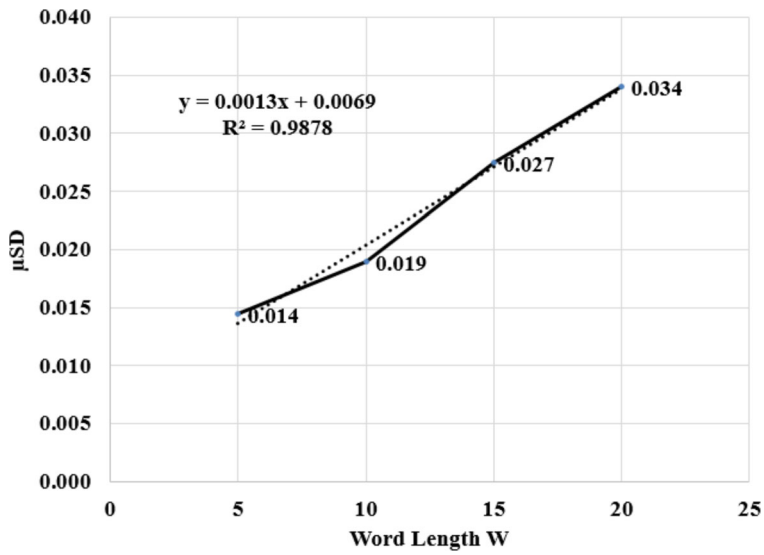

Fig. 25 Relation between the average prediction accuracy and the length of the predicted *W*

The second objective is to study the effectiveness of the proposed CumP-validation procedure under the concept of dividing the whole prediction time into subintervals compared to the common one (i.e, CrossP). Results show that the CumP provides nearly the same accuracy of the cross-validation with 57% off computational time.

The third objective is to study the effectiveness of using clustering techniques (i.e., K-means) as a discretizing technique combined with filtering techniques (i.e., SG) in the data preparation phase. Results show the effectiveness of using k-means instead of the binning technique for discretization.

According to the results, considering work-load prediction as a translation problem might open the door for using other translation techniques. So, studying other machine translation techniques should be considered as future work. Also, evaluating the proposed AS2S technique and the CumP procedure using a huge volume of time-series training datasets should be considered as future work.

Data Availability Statement (DAS) Experiments included in this paper have been conducted on the total CPU and memory usage, so the average CPU usages and memory usages of all tasks on every machine at every timestamp were calculated and stored in a text file with a name "the machine ID". Therefore, 790 text files have been generated and organized in records (one for each 300 s) of the order "timestamp, CPU, and Memory". These files are available in [85] repository.

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Declarations

Conflict of Interest There is no conflict of interest.

References

1. Al-Sayed, M.M., Hassan, H.A., Omara, F.A.: Towards evaluation of cloud ontologies. J. Parallel Distrib. Comput. **126**, 82–106 (2019)
2. Al-Sayed, M.M., Hassan, H.A., Omara, F.A.: CloudFNF: an ontology structure for functional and non-functional. J. Parallel Distrib. Comput. **14**, 143–173 (2020)
3. Al-Sayed, M.M., Hassan, H.A., Omara, F.A.: Mapping lexical gaps in cloud ontology using BabelNet and FP-growth. Int. J. Comput. Sci. Secur. (IJCSS). **13**(2), 36–52 (2019)
4. Zharikov, E., Telenyk, S., Bidyuk, P.: Adaptive workload forecasting in cloud data centers. J. Grid Comput. **18**(1), 149–168 (2020)
5. Chen, Z., Hu, J., Min, G., Zomaya, A.Y., El-Ghazawi, T.: Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning. IEEE Trans. Parallel Distrib. Syst. **31**(4), 923–934 (2020)
6. Amiri, M., Mohammad-Khanli, L.: Survey on prediction models of applications for resources provisioning in cloud. J. Netw. Comput. Appl. **82**, 93–113 (2017)
7. Dabbagh, M., Hamdaoui, B., Guizani, M., Rayes, A.: Energy-efficient resource allocation and provisioning framework for cloud data centers. IEEE Trans. Netw. Serv. Manag. **12**(3), 377–391 (2015)
8. Kumar, J., Goomer, R., Singh, A.K.: Long short term memory recurrent neural network (LSTM-RNN) based workload forecasting model for cloud datacenters. Procedia Comput. Sci. **125**, 676–682 (2018)
9. Dabbagh, M., Hamdaoui, B., Guizani, M., Rayes, A.: An energy-efficient vm prediction and migration framework for overcommitted clouds. IEEE Trans. Cloud Comput. **6**(4), 955–966 (2016)
10. Kumar, J., Singh, A.K., Buyya, R.: Self directed learning based workload forecasting model for cloud resource management. Inf. Sci. **543**, 345–366 (2020)
11. GUREYA, D.D.: Resource Allocation for Data-Intensive Services in the Cloud. Universitetsservice US-AB, Stockholm (2021)
12. Schmidt, T., Marg, L.: How to Move to Neural Machine Translation for Enterprise-Scale Programs—An Early Adoption Case Study. In: Proceedings of the 21st Annual Conference of the European Association for Machine Translation, Alacant, Spain 2018
13. Al-Sayed, M.M., Khattab, S., Omarab, F.A.: Prediction mechanisms for monitoring state of cloud resources using Markov chain model. J. Parallel Distrib. Comput. **96**, 163–171 (2016)
14. Amiri, M., Mohammad-Khanli, L., Mirandola, R.: An online learning model based on episode mining for workload prediction in cloud. Futur. Gener. Comput. Syst. **87**, 83–101 (2018)
15. Saxena, D., Chauhan, R., Kait, R.: Dynamic fair priority optimization task scheduling algorithm in cloud computing: concepts and implementations. Int. J. Comput. Netw. Inf. Secur. **8**(2), 41–48 (2016)
16. Saxena, D., Singh, A.K.: Communication cost aware resource efficient load balancing (care-lb) framework for cloud datacenter. Recent Adv. Comput. Sci. Commun. **12**, 1–14 (2020)

17. Zhang, F., Liu, G., Fu, X., Yahyapour, R.: A survey on virtual machine migration: challenges techniques and open issues. IEEE Commun. Surv. Tutor. **20**(2), 1206–1243 (2018)
18. Kumar, J., Singh, A.K.: Workload prediction in cloud using artificial neural network and adaptive differential evolution. Futur. Gener. Comput. Syst. **81**, 41–52 (2018)
19. Kirchoff, D.F., Xavier, M., Mastella, J., De Rose, C.A.: A Preliminary Study of Machine Learning Workload Prediction Techniques for Cloud Applications. In: 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Pavia, Italy, Italy Feb. 2019
20. Tang, X.: Large-scale computing systems workload prediction using parallel improved LSTM neural network. IEEE Access. **7**, 40525–40533 (2019)
21. Singh, P., Gupta, P., Jyoti, K.: TASM: technocrat ARIMA and SVR model for workload prediction of web applications in cloud. Clust. Comput. **22**, 619–633 (2019)
22. Kim, I.K., Wang, W., Qi, Y., Humphrey, M.: Forecasting cloud application workloads with cloudinsight for predictive resource management. IEEE Trans. Cloud Comput (Early Access). 1–16 (2020)
23. Kumar, J., Singh, A.K., Buyya, R.: Self directed learning based workload forecasting model for cloud resource management. Inf. Sci. **543**, 345–366 (2021)
24. Daraghmeh, M., Melhem, S.B., Agarwal A., Goel, N., Zaman, M.: Regression-Based Dynamic Provisioning and Monitoring for Responsive Resources in Cloud Infrastructure Networks. In: IEEE/ACS 15th International Conference on Computer Systems and Application (AICCSA), Jordan (2018)
25. Al-Dhuraibi, Y., Paraiso, F., Djarallah, N., Merle, P.: Elasticity in cloud computing: state of the art and research challenges. IEEE Trans. Serv. Comput. **11**(2), 430–447 (2018)
26. Daraghmeh, M., Agarwal, A., Manzano, R., Zaman, M.: Time Series Forecasting Using Facebook Prophet for Cloud Resource Management. In: IEEE International Conference on Communications Workshops (ICC Workshops), Canada (2021)
27. de Nardin, I.F., da Righi, R.R., Lopes, T.R., Costa, C., YoungYeom, H.: On revisiting energy and performance in microservices applications: A cloud elasticity-driven approach. Parallel Comput. **108**, 102858 (2021)
28. Dabbagh, M., Hamdaoui, B., Guizani, M., Rayes, A.: Exploiting task elasticity and price heterogeneity for maximizing cloud computing profits. IEEE Trans. Emerg. Top. Comput. **6**(1), 85–96 (2015)
29. Malinowski, K., Mande, J.: Lionbridge Translation Technologies. [Online]. Available: https://www.lionbridge.com/blog/translation-localization/the-future-of-language-technology-the-future-of-machine-translation/ (23 12 2021). Accessed 12 2021
30. Dai, H.: Comparative Analysis of Machine Translation and Human Translation under the Background of Internet. In: International Conference on Cognitive Based Information Processing and Applications (CIPA), Singapore (2021)
31. Choudhary, H., Pathak, A.K., Shah, R.R., Kumaraguru, P.: Neural Machine Translation for English-Tamil. In: Proceedings of the Third Conference on Machine Translation (WMT), Brussels, Belgium (2018)
32. Yang, J., Yin, Y., Ma, S., Huang, H., Zhang, D., Li, Z., Wei, F.: Multilingual agreement for multilingual neural machine translation. In: Proceedings of the 59th annual meeting of the Association for Computational Linguistics and the 11th international joint conference on natural language processing (ACL-IJCNLP), Online (2021)
33. Cromieres, F., Nakazawa, T., Dabre, R.: Neural Machine Translation: Basics, Practical Aspects and Recent Trends. In: Proceedings of the the 8th International Joint Conference on Natural Language Processing (IJCNLP), Taiwan (2017)
34. Guo, J., Chang, Z., Wang, S., Ding, H., Feng, Y., Mao, L., Bao, Y.: Who Limits the Resource Efficiency of my Datacenter: an Analysis of Alibaba Datacenter Traces. In: IEEE/ACM 27th International Symposium on Quality of Service (IWQoS), Phoenix, AZ, USA, USA (2019)
35. Zhu, Y., Zhang, W., Chen, Y., Gao, H.: A novel approach to workload prediction using attention-based LSTM encoder-decoder network in cloud environment. EURASIP J. Wirel. Commun. Netw. **274**, 1–18 (2019)
36. Hannan, E.J., Kavalieris, L.: REGRESSION, AUTOREGRESSION MODELS. J. Time Ser. Anal. **7**(1), 27–49 (1986)
37. Robinson, P.M.: The estimation of a nonlinear moving average model. Stoch. Process. Appl. **5**(1), 81–90 (1977)
38. Nelson, B.K.: Time series analysis using autoregressive integrated moving average (ARIMA) models. Acad. Emerg. Med. **5**(7), 739–744 (1998)
39. Bernardo, J.M., Smith, A.F.: Bayesian theory, 1st edn. New York: John Wiley and Sons (2009)
40. Peterson, L.E.: K-nearest neighbor. Scholarpedia. **4**(2), 1883 (2009)
41. Alpaydin, E.: Introduction to Machine Learning, 4th edn. MIT Press, United States (2020)
42. Bergmeir, C., Benítez, J.M.: On the use of cross-validation for time series predictor evaluation. Inf. Sci. **191**, 192–213 (2012)
43. Tashman, L.J.: Out-of-sample tests of forecasting accuracy: an analysis and review. Int. J. Forecast. **16**(4), 437–450 (2000)
44. Singh, G.B.: Fundamentals of Bioinformatics and Computational Biology, vol. 6, pp. 97–125. Springer International Publishing, Switzerland (2015)
45. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
46. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. Adv. Neural Inf. Proces. Syst. **27**, 3104–3112 (2014)
47. Goyal, P., Pandey, S., Jain, K.: Deep Learning for Natural Language Processing: Creating Neural Networks with Python. Apress, Berkeley (2018)
48. Schafer, R.W.: On the Frequency-Domain Properties of Savitzky-Golay Filters. In: Digital Signal Processing and Signal Processing Education Meeting (DSP/SPE), USA 2011
49. Farahnakian, F., Liljeberg, P., Plosila, J.: LiRCUP: Linear Regression Based CPU Usage Prediction Algorithm for Live Migration of Virtual Machines in Data Centers. In: 39th Euromicro Conference on Software Engineering and Advanced Applications, Santander, Spain (October 2013)
50. Islam, S., Keung, J., Lee, K., Liu, A.: Empirical prediction models for adaptive resource provisioning in the cloud. Futur. Gener. Comput. Syst. **28**(1), 155–162 (2012)

51. Singh, S., Chana, I.: A survey on resource scheduling in cloud computing: issues and challenges. Journal of Grid Computing. **14**(2), 217–264 (2016)
52. Chen, X., Wang, H., Ma, Y., Zheng, X., Guo, L.: Self-adaptive resource allocation for cloud-based software services based on iterative QoS prediction model. Futur. Gener. Comput. Syst. **105**, 287–296 (2020)
53. Yoo, W., Sim, A.: Time-series forecast modeling on high-bandwidth network measurements. J. Grid Comput. **14**, 463–476 (2016)
54. Naseera, S., Rajini, G.K., Prabha, N.A., Abhishek, G.: A comparative study on CPU load predictions in a computational grid using artificial neural network algorithms. Indian J. Sci. Technol. **8**(35), 1–5 (2015)
55. Yadav, M.P.; Pal, N.; Yadav, D.K.: Workload Prediction over Cloud Server Using Time Series Data. In: 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India (2021)
56. Désiré, K.K., Francis, K.A., Kouassi, K.H., Dhib, E., Tabbane, N., Asseu, O.: Fractional rider deep long short term memory network for workload prediction-based distributed resource allocation using spark in cloud gaming. Engineering. **13**, 135–157 (2021)
57. Bi, J., Li, S., Yuan, H., Zhou, M.C.: Integrated deep learning method for workload and resource prediction in cloud systems. Neurocomputing. **424**, 35–48 (2021)
58. Xue, J., Yan, F., Birke, R., Chen, L.Y., Scherer, T., Smirni, E.: PRACTISE: Robust Prediction of Data Center Time Series. In: 11th International Conference on Network and Service Management (CNSM), Barcelona, Spain (2015)
59. Box, G.E.P., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: Time Series Analysis: Forecasting and Control, 5th edn. Wiley, Hoboken (2015)
60. Hayashi, S., Tanimoto, A., Kashima, H.: Long-Term Prediction of Small Time-Series Data Using Generalized Distillation. In: International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, Hungary (July 2019)
61. Taieb, S.B.; Bontempi, G., Sorjamaa, A., Lendasse, A.: Long-Term Prediction of Time Series by Combining Direct and MIMO Strategies. In: International Joint Conference on Neural Networks, Atlanta, GA, USA (June 2009)
62. Mathworks documentation team, "Sliding Window Method and Exponential Weighting Method," The Mathworks, (1994-2022). [Online]. Available: https://www.mathworks.com/help/dsp/ug/sliding-window-method-and-exponential-weighting-method.html#:~:text=In%20the%20sliding%20window%20method,the%20Len%20%2D%201%20previous%20samples. Accessed 29 Apr 2022
63. Tang, X., Liao, X., Zheng, J.: Energy efficient job scheduling with workload prediction on cloud data center. Clust. Comput. **21**(3), 1581–1593 (2018)
64. Tosun, E., Aydin, K., Bilgili, M.: Comparison of linear regression and artificial neural network model of a diesel engine fueled with biodiesel-alcohol mixtures. Alex. Eng. J. **55**(4), 3081–3089 (2016)
65. Zuur, A.F., Ieno, E.N., Walker, N.J., Saveliev, A.A., Smith, G.M.: Limitations of Linear Regression Applied on Ecological Data. In: Mixed Effects Models and Extensions in Ecology with R. Statistics for Biology and Health, New York, Springer, pp. 11–33 (2009)
66. Amiri, M., Mohammad-Khanli, L., Mirandola, R.: A sequential pattern mining model for application workload prediction in cloud environment. J. Netw. Comput. Appl. **105**, 21–62 (2018)
67. Amiri, M., Mohammad-Khanli, L., Mirandola, R.: A new efficient approach for extracting the closed episodes for workload prediction in cloud. Computing. **102**, 141–200 (2020)
68. Amiri, M., Feizi-Derakhshi, M., Mohammad-Khanli, L.: IDS fitted Q improvement using fuzzy approach for resource provisioning in cloud. J. Intell. Fuzzy Syst. **32**(1), 229–240 (2017)
69. Wang, Z., Hong, T.: Reinforcement learning for building controls: the opportunities and challenges. Appl. Energy. **269**, 115036 (2020)
70. Bi, J., Yuan, H., Zhou, M.: Temporal prediction of multiapplication consolidated workloads in distributed clouds. IEEE Trans. Autom. Sci. Eng. **16**(4), 1763–1773 (2019)
71. Sovic, A., Sersic, D.: Signal decomposition methods for reducing drawbacks of the DWT. Eng. Rev. **32**(2), 70–77 (2012)
72. Kumar, J., Singh, A.K.: Performance evaluation of metaheuristics algorithms for workload prediction in cloud environment. Appl. Soft Comput. **113**(Part A), 1–14 (2021)
73. Roodschild, M., Sardiñas, J.G., Will, A.: A new approach for the vanishing gradient problem on sigmoid. Prog. Artif. Intell. **9**(4), 351–360 (2020)
74. Wang, X., Qin, Y., Wang, Y., Xiang, S., Chen, H.: ReLTanh: an activation function with vanishing gradient resistance for SAE-based DNNs and its application to rotating machinery fault diagnosis. Neurocomputing. **363**, 88–98 (2019)
75. Bahdanau, D., Cho, K., Bengio, Y.: NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE. In: arXiv preprint arXiv:1409.0473 (2015)
76. Chitrakar, R., Chuanhe, H.: Anomaly Detection Using Support Vector Machine Classification with K-Medoids Clustering. In: 3rd Asian Himalayas International Conference on Internet, Kathmundu, Nepal (2013)
77. Savitzky, A., Golay, M.J.: Smoothing and differentiation of data by simplified least squares procedures. Anal. Chem. **36**(8), 1627–1639 (1964)
78. Luo, J., Ying, K., He, P., Bai, J.: Properties of Savitzky-Golay digital differentiators. Digital Signal Processing. **15**(2), 122–136 (2005)
79. Sadeghi, M., Behnia, F., Amiri, R.: Window selection of the Savitzky–Golay filters for signal recovery from Noisy measurements. IEEE Trans. Instrum. Meas. **69**(8), 5418–5427 (2020)
80. Brandl G. "pandas.cut," pandas development team (2008-2022). [Online]. Available: https://pandas.pydata.

org/docs/reference/api/pandas.cut.html. Accessed 28 Aug 2021
81. Reiss, C., Tumanov, A.; Ganger, G.R., Katz, R.H.: Towards understanding heterogeneous clouds at scale: Google trace analysis. *Intel science and Technology Center for Cloud Computing*, Tech. Rep 84 (2012)
82. Chaves, S.A.D., Uriarte, R.B., Westphall, C.B.: Toward an architecture for monitoring private clouds. IEEE Commun. Mag. **49**(12), 130–137 (2011)
83. Zou, F., Shen, L., Jie, Z., Zhang, W., Liu, W.: A Sufficient Condition for Convergences of Adam and RMSProp. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA (2019)
84. Zhang, P., Zhang, H., Dai, J., Yi, Y., Zhang, H., Zhang, Y.: Deep learning on computational-resource-limited platforms: a survey. Mob. Inf. Syst. **2020**, 1–19 (2020)
85. Al-Sayed, M.M.: Google Dataset (2022). [Online]. Available: https://drive.google.com/file/d/1p4Y3LitdBCIE0YZHvGFNJLG8ey72hXHH/view?usp=sharing. Accessed 29 Apr 2022