CANFIS: A Chaos Adaptive Neural Fuzzy Inference System for Workload Prediction in the Cloud

ZOHRA AMEKRAZ1, MOULAY YOUSSEF HADI1
1Laboratory of Information Modeling and Communication Systems, Faculty of Sciences, Ibn Tofail University, BP 133, Kenitra, Morocco
Corresponding author: Zohra Amekraz (e-mail: zohra.amekraz@gmail.com).

ABSTRACT
As cloud-based applications become increasingly solicited by companies and individuals, the competition between cloud providers that offer cloud services keeps increasing. To win this competition, cloud providers must provide sufficient computing resources that will satisfy users’ demand for every request. Workload prediction has been investigated extensively to resolve this issue using various techniques. This paper presents a workload prediction method called CANFIS, which combines the Savitzky-Golay (SG) filter and Chaotic time series analysis with the Adaptive Neural Fuzzy Inference System (ANFIS) to make predictions of cloud workloads. The SG filter is used to clean data from noise and outliers, and chaotic analysis is used to investigate the chaotic nature of workload and to build the improved ANFIS model. The proposed method is evaluated using real workload traces from web applications (i.e., Wikipedia and NASA Kennedy traces) and cluster applications (i.e., CPU and Memory of Google cluster). Experimental results show that the proposed CANFIS model can improve prediction accuracy compared to existing techniques, including simple ANFIS, Auto-Regressive Integrated Moving Average (ARIMA), Support Vector Machine (SVM), Long Short Term Memory (LSTM), and Neural Networks based methods. A statistical analysis is also performed using the Friedman test, along with Finner post hoc analysis to verify the efficiency of the proposed prediction model.

INDEX TERMS Adaptive Neuro-Fuzzy Inference System, chaos analysis, cloud application workloads, proactive technique, workload prediction.

I. INTRODUCTION

CLOUD computing is a powerful paradigm that has been evolving in the computing field for the last few years. It allows individuals and enterprises to deploy their applications, store their data and obtain access to various resources deployed in data centers that are held and managed by third-party cloud providers. All resources in the cloud are offered as on-demand services to users on a pay-as-you-go basis [1].

The number of companies and individuals willing to use cloud services increases every day due to the various advantages of the cloud, such as elasticity and on-demand scalability. However, due to this quick growth in the number of cloud users, some challenges related to efficient resource management in data centers have begun to arise. Cloud computing providers have to provide sufficient resources to manage the large and complex demands of users; otherwise, the Service Level Agreement (SLA) terms will be violated, and cloud providers are likely to lose their customers. Over-provisioning resources is not the best solution, particularly when a period of low demand occurs because in that case, the excess of available resources will remain idle, wasting resources [2]. One possible solution to these under- and over-provisioning issues is to use prior knowledge of upcoming demand. Cloud providers can use this prior knowledge to perform proactive provisioning and scheduling of resources and consequently avoid SLA violations and maximize resource utilization [3].

A. RESEARCH MOTIVATION AND CHALLENGES
As much as the prediction of cloud workload is the solution to the encountered problems in cloud resource management, it is a challenge by itself. The first major challenge in workload prediction is the presence of nonlinearity in workloads [4]; for example, demand at a given time interval could be on the order of millions of requests but zero or on the order...
of ten at the next time interval, which results in unexpected peak and slack periods in workload patterns. These sudden fluctuations in workload provoke under- or overestimation of required resources, which results in violation of SLA terms, extreme power consumption and resource wastage [5]. The second major challenge in workload prediction is the presence of nonrepeating patterns [6], [7]; for example, some patterns seem to be indistinguishable on regular workdays with the presence of peaks in the morning as well as in the afternoon and low demand at midnight, but they are not exactly the same. Therefore, finding a prediction method that can consider the aforementioned characteristics of cloud workloads can play an important role in improving the prediction accuracy.

B. PROPOSED APPROACH

In this paper, we propose a prediction model that combines chaotic time series analysis and the ANFIS model to predict cloud workload.

First, chaotic time series analysis is used to show that the workload system has a chaotic nature. Chaos is one form of behavior displayed in nonlinear systems and is used to characterize the system’s time behavior when that behavior does not exactly repeat itself (i.e., aperiodic) and appears to be noisy [8]. Once chaos is identified in the cloud workload, we can say that workload time series are nonlinear, deterministic and dynamic, and can be predicted on a short-term scale [9]. Therefore, we can use a nonlinear model to perform a short-term prediction of these chaotic workloads.

The ANFIS model [10] is an intelligent model that has the ability to approach complex nonlinear systems and has been explored extensively in the prediction of cloud workloads [11] - [13]. The good accuracy achieved by this model is encouraging. Thus, in this paper, we also use the ANFIS model to predict chaotic cloud workloads. To overcome the limitation of the ANFIS model regarding the selection of its inputs, this paper suggests the use of methods of chaotic time series analysis, the phase space reconstruction theory, to select these inputs.

C. CONTRIBUTIONS

In brief, the major contributions of this paper are as follows:

- To clean data from possible outliers and noise while preserving the width and the peak of the signal, we use a smoothing method called the Savitzky-Golay filter [14].
- We examine the existence of chaos in cloud workloads using chaotic time series analysis.
- We predict chaotic workloads using an ANFIS model.
- We evaluate the performance of the proposed prediction method using various real workload traces collected from web and cluster applications.

D. ORGANIZATION OF THE PAPER

The remainder of this paper is structured as follows. The related work is given in Section II. In Section III, the proposed solution is explained in detail. Section IV evaluates the proposed prediction model using various real workload traces. Finally, Section VI concludes the paper.

II. RELATED WORK

Workload prediction is a challenging issue that has attracted the attention of researchers in cloud computing for the last few years. This is due to its utility in the development of efficient resource provisioning strategies that can improve the overall performance of the cloud system. Many studies have investigated this issue, and each uses a different category of prediction model.

Amiri et al. [15] presented an exhaustive review of the application prediction methods in various aspects. They proposed a taxonomy for prediction methods that can examine the principal aspects and challenges of these methods. For statistical-based prediction models, Calheiros et al. [16] applied an ARIMA model to dynamically forecast and provisioning resources for SaaS providers. In [17], a seasonal ARMA model was used to forecast workload data up to 168 h forward using expert guidance for order optimization purposes. Roy et al. [18] predicted the workload using moving average, exponential moving average and ARMA models. Amekraz et al. [19] presented a strategy for ARMA model selection based on the Gaussianity feature of the workload. If the workload was non-Gaussian, a higher order ARMA model was applied, while if the workload was Gaussian, the second order ARMA model was used.

Regarding pattern matching-based techniques, Caron et al. [20] searched for patterns in the archival data that are similar to the current pattern to make predictions of the workload. In [21], the authors proposed an Euclidian distance-based string pattern matching algorithm to forecast the workload.

For machine learning-based models, Kumar et al. [22] proposed an artificial neural network based on self-adaptive differential evolution (SaDE) and showed that the proposed model outperformed the backpropagation trained neural network model of [23]. Cao et al. [24] used machine learning methods such as random forest to forecast server load using monitoring of historical data and showed that these methods outperform traditional time series methods in terms of prediction accuracy. In [4], a workload prediction framework was presented that combines a neural network model and supervised learning. An improvement of the differential evolution algorithm was performed to ameliorate the efficiency of learning of the prediction model. In [25], the authors presented a workload forecasting method based on the black-hole algorithm. The latter was used to train the neural network model. In [26], the authors proposed a self-directed workload forecasting method (SDWF) that captures the prediction error trend by calculating the deviation in recent predictions and used it to ameliorate the accuracy of future predictions. The authors developed an improved black-hole algorithm for the training of network neurons to achieve more accurate predictions. In [27], the authors proposed an LSTM neural network method that uses a two-dimensional time series instead of one to train the LSTM model. Singh
### TABLE 1: Review of works related to cloud workload prediction techniques

| Ref  | Prediction technique                                           | Workload dataset                                      | Evaluation tools                  | Performance metrics                     |
|------|---------------------------------------------------------------|-------------------------------------------------------|-----------------------------------|-----------------------------------------|
| [4]  | NN + Improved adaptive differential evolution algorithm       | Nasa, Saskatchewan, and Google cluster                | Jupyter notebook                  | RMSE, Training time, and Convergence speed |
| [11] | Ensemble model + Subtractive fuzzy clustering + Fuzzy Neural Network | Network traffic data                                  | NA                                | MAE, MSE, and PRED                      |
| [16] | ARIMA                                                         | Wikipedia                                             | Cloudsim                          | RMSD, NRMSD, MAD, and MAPE              |
| [17] | Seasonal ARMA                                                 | achat-ville.com                                      | Java                              | MAPE                                    |
| [19] | Higher order statistics + ARMA model                         | Wikipedia                                             | Matlab                             | MAPE and NRMSE                          |
| [20] | String matching                                              | Animoto, LCG, NorduGrid, and SHARCNET                | NA                                | UCSB metrics, minimum, maximum, median, and average percentage prediction error and Average runtime |
| [21] | String matching                                              | Words count program, Computational tasks, Web applications, and memory consumption applications. | Aliyun                             | MAPE                                    |
| [22] | NN + Self adaptive Differential Evolution algorithm (SaDE)    | NASA and Saskatchewan                                | Matlab                             | RMSE, Convergence speed, and Training time |
| [26] | NN + Improved black-hole algorithm (SDWF)                    | NASA, Saskatchewan, Google cluster, and PlanetLab    | Matlab + SPSS                      | MSE, MAE, and MAPE                      |
| [27] | Improved LSTM                                                | Shanghai Supercomputer Center                        | NA                                | MSE and speedup                          |
| [28] | Evolutionary Quantum Neural Network (EQNN)                   | Google cluster, PlanetLab, Nasa, NorduGrid, and SharCNet | Phyton + STAC web platform        | RMSE and MAE                            |
| [29] | ANFIS                                                         | Synthetic, and RuneScape                              | Matlab                             | MSE, RMSE, \( R^2 \)                    |
| [32] | Sequential pattern mining                                    | Synthetic workloads and Bitbrain                      | NA                                | Prediction precision                     |
| [34] | Bidirectional LSTM + Gated Recurrent Unit                    | Bitbrains                                             | Google Colaboratory                | MSE, RMSE, MAE, MAPE                    |
| [36] | Base predictors (AR and Gray model) + Deep belief network + Particle swarm optimization | Google cluster                                      | Matlab                             | MAE, MSE, MAPE, and error               |
| [37] | Ensemble model : AR, ARIMA, LR, SYM...                        | Google cluster, Facebook, Grid5000, NorduGrid, AuverGrid, SHARCNet, and LCG | Python                             | RMSE                                   |
| [38] | SVM and LR                                                    | Google cluster                                       | NA                                | MRPE and Cumulative Relative Error      |
| [39] | Learning Automata theory                                      | PlanetLab                                             | R software                         | Execution time, RMSD, Error Ratio, and Absolute Error |
| [40] | Bayesian information + Smooth filters                         | PlanetLab                                             | R software                         | MAE, RMSE, MAPE, and MASE               |

**Proposed** SG filter + Chaotic time series analysis + ANFIS | Wikipedia, Nasa, and Google cluster | Matlab + STAC web platform | RMSE, MAE, MAPE, and \( R^2 \) |
TABLE 2: Advantages and disadvantages of the related works

| Ref       | Advantages                                  | Disadvantages                                           |
|-----------|---------------------------------------------|---------------------------------------------------------|
| [4]       | Faster convergence compared to self adaptive differential evolution algorithm | Higher complexity than self adaptive differential evolution algorithm |
| [11]      | Decrease the amount of calculation          | May increase time delay and Execution time              |
| [16], [17]| Simplicity                                  | Needs to be retrained to adapt to workload changes + Unsuitable for nonlinear and non Gaussian workloads |
| [19]      | Suitable for non Gaussian workloads         | Needs to be retrained to adapt to workload changes + Unsuitable for nonlinear and non Gaussian workloads |
| [20], [21]| Fast                                       | Needs preprocessing + Selection of the length of pattern |
| [22]      | Accuracy improvement compared to Back-propagation | Static prediction approach                              |
| [26]      | Accuracy improvement compared to Black-hole, SaDE, and Back-propagation | Higher time complexity                                   |
| [27]      | Accuracy improvement compared to LSTM in large-scale computing systems | No information whether it is convenient to predict CPU or other workloads |
| [28]      | Higher and stable prediction accuracy       | Higher complexity due to qubits generation and processing |
| [29]      | Accuracy improvement                        | Selection of model parameters                            |
| [32]      | No assumptions about the behavior of the workload | Inability to adapt to the workload changes.  |
| [34]      | Accuracy improvement compared to ARIMA, LSTM, GRU, and Bi-LSTM | High training time                                      |
| [36]      | Accuracy improvement                        | Time consuming                                           |
| [37]      | Real-time learning and acceptable accuracy  | Higher complexity than single machine learning models    |
| [38]      | Accuracy improvement                        | Complexity increasing                                    |
| [39]      | Accuracy improvement and less execution time compared to Genetic algorithm | Prediction of one type of resources (i.e., CPU) |
| [40]      | Simplicity + Accuracy improvement compared to Genetic algorithm and Ensemble model | Inability to adapt to the workload changes |

et al. [28] presented an evolutionary quantum neural network (EQNN) for workload prediction. The method uses quantum computing to encode workload information into qubits. The weight of the qubits is adjusted using the rotation effects of the controlled-NOT gate serve activation function. Also, the qubit weights are optimized using the self-balanced adaptive differential evolution (SB-ADE) algorithm. In [29], the authors used an ANFIS model to forecast future workload and a fuzzy decision tree algorithm to determine the suitable number of resources to allocate in the Massively Multiplayer Online Games applications. The results of simulations indicate that their proposed approach outperforms existing methods in terms of accuracy and performance. Amiri et al. [30] presented a provisioning method based on workload prediction. The prediction was performed using a neural network model, while the estimation of the number of physical machines was performed using the Q learning technique. The results of simulations show that their proposal had an important effect on the convergence speed and decreases the learning time. In [31], the authors presented a new Clustered Induced Ordered Weighted Averaging (IOWA) Adaptive Neuro-Fuzzy Inference System (ANFIS) model to predict QoS. Their proposed model could reduce the dimension of data and manage the nonlinear relationship of the QoS dataset. For sequential pattern mining-based prediction models, Amiri et al. [32] presented for the first time a sequential pattern mining-based prediction method to forecast future cloud demand. This method, called POSITING, could extract all behavioral patterns of workloads by investigating the correlation between resources without making assumptions about the behavior of the workload. Therefore, the proposed model could be used for various types of workloads. Although the proposed model yields good results, it cannot adapt to workload changes. Thus, in [5], the authors suggested a forecasting model that combines episode mining with online learning (RELENTING) to ameliorate POSITING. Their model was supposed to fulfill the adaptability characteristic of the prediction models along with accuracy. To improve the space complexity of the pattern mining technique, Amiri et al. [33] defined redundant occurrences of patterns and proved that eliminating them does not cause any loss of information. Using PROSPER, which is an imProved RepresentatiOn of the Stream based on PointERs, the authors developed the algorithms in [32] to identify redundant occurrences and eliminate them. In addition to reducing time complexity, the authors proposed a solution to improve the space complexity of the pattern mining technique. The suggested solution consists of a data structure called the closed pattern backward tree (CPBT), which can extract and store closed patterns directly without processing or storing all the possible closed
patterns.

A combination of two or more of the previously mentioned models (i.e., statistical, machine learning models...) is also widely used by several researchers. Notably, Karim et al. [34] presented a hybrid prediction model called BHyPreC, which is based on a recurrent neural network, to predict the CPU usage workload of a cloud’s virtual machine. The proposed model includes a gated recurrent unit (GRU) and bidirectional long short-term memory (LSTM). Gong et al. [35] presented a scaling mechanism called PRESS that uses fast Fourier transform (FFT) to extract repeating patterns that will be used during workload predictions. If no repeating patterns are found, PRESS uses a statistical state-driven technique and employs a Markov chain to forecast demand. Wen et al. [36] used particle swarm optimization with a deep belief network (DBN) to predict CPU usage. After preprocessing the data, the AR and gray models are used as base prediction methods and trained to give additional input information for the DBN.

Regarding ensemble model-based prediction approaches, Kim et al. [37] used an ensemble model based on many predictors to make forecasts for real-world workloads. The authors used a total of eight predictors in their model, including robust stepwise linear regression (RSLR), support vector machine (SVM), and ARIMA models. In [38], the authors made use of workload characteristics, including burstiness, to select a convenient workload forecasting model among linear regression (LR) and SVM models. Rahmanian et al. [39] suggested a prediction approach for CPU utilization that uses a combination of many baseline prediction models. The method uses the theory of learning automata to compute the weight for each of the prediction models used. Tofighy et al. [40] presented an ensemble model for load prediction based on the Bayesian information criterion (BIC) and smooth filters. The BIC is used to select the best element model in every time slot using a history of resource usage. Smooth filters are used to reduce the negative effect of outliers in the data. Ghobaei-Arani [41] proposed a workload classification method that combines a biogeography-based optimization (BBO) technique with K-means clustering to classify workloads according to QoS needs. Also, the author used the Bayesian learning method to decide proper resource provisioning operations to meet the requirements of the QoS. Etemadi et al. [42] proposed a Bayesian learning-based provisioning technique for fog computing that enables dynamic scaling of resources based on the predicted workload. Workload prediction is then performed using several prediction models, including the MA, ARMA, ARIMA, and AR models. The appropriate model that will be used to make predictions corresponds to the model with the minimum error values. Evaluation results demonstrated that their proposed method could reduce the cost and delay violations and increase the use of fog nodes. Chen et al. [11] proposed an adaptive forecasting method that uses an improved fuzzy neural network model. The latter was constructed using an ensemble model and subtractive fuzzy clustering. Although ensemble model-based approaches are effective in enhancing the accuracy of workload prediction, they are time-consuming compared to single-method approaches.

In the proposed approach, we use the SG filter to clean the historical data and the Neuro-Fuzzy Inference System with chaos theory (CANFIS) to forecast the future demand. Chaos theory provides a new and different perspective, from which we can visualize, analyze and understand workloads in the cloud.

Table 1 summarizes relevant works related to workload prediction in the cloud based on four features: (1) prediction technique, (2) workload dataset, (3) evaluation tools, and (4) performance metrics. Also, Table 2 describes some of the advantages and disadvantages of these related works.

III. PROPOSED PREDICTION STRATEGY

This section presents the proposed workload prediction model. First, the general architecture of the cloud resource management system is introduced, and the different interactions between its primary entities are briefly described. Next, the structure of the proposed workload predictor is explained in detail. Finally, the time complexity of the proposed predictor is given.

A. CLOUD RESOURCE MANAGEMENT SYSTEM

In this section, we present the global architecture of the cloud resource management system. The primary components of this architecture are shown in Figure 1. The different interactions between these components are shown in Figure 2.

As shown in Figure 1, the cloud resource management system consists of five major modules: the admission controller, the data warehouse, the monitor, the workload predictor, and the resource manager module. We will now explain the functionalities of these modules as well as the different interactions between them.

At the beginning of each time slot, the monitor module collects information about requests (e.g., amount of requests,
response time, etc.) and the state of the computing resources (e.g., CPU, RAM, storage, etc.) from the cloud infrastructure. The collected information is stored in the data warehouse. Then, this information is fed into the workload predictor module to predict future demand. The contributions of this paper concern the workload predictor module, where the proposed structure will be explained in detail in the next section. The results of the prediction given by the workload predictor module are sent to the resource manager module. The resource manager uses the information about the state of the computing resources that are stored in the data warehouse and the prediction results to determine the scaling decisions (i.e., scale up, scale down, or no action) that should be taken. Next, the resource manager transfers the scaling decisions to the cloud infrastructure and makes the required changes. Also, the resource manager module will dispatch the accepted user requests received from the admission controller to the appropriate VMs according to different strategies (i.e., round robin, first fit, etc.). Finally, the users’ requests are executed by the selected VMs.

B. PROPOSED WORKLOAD PREDICTION METHOD

This section presents the proposed workload prediction method. Primarily, the method consists of three steps, as shown in Figure 3. First, historical data are processed using a set of preprocessing operations, including extraction and aggregation, cleaning with an SG filter, and normalization. The resulting processed workload time series are fed to the chaotic time series analysis module to identify the chaotic characteristics of the workload. This analysis consists of reconstructing the phase space and calculating the MLE to verify the chaotic nature of the workload. Finally, the inputs of the CANFIS model are selected based on the chaotic study that is performed in the previous module. The CANFIS model is then built, trained and tested, and the prediction results are forwarded to the resource manager. A detailed explanation of every part of the proposed workload prediction method is given in the rest of this section.
Analysis of Cloud of MLE time series workloads and CPU usage and memory usage for cluster time intervals.

During this step, the desired fields in the historical processing steps. The first step is data extraction and aggregation. In this paper, we use the SG filter to clean data from abnormal events, such as machine failures [43]. Therefore, parts are likely to be contaminated by noise caused by some possible outliers and noise.

In this paper, we focus on the number of requests for web workloads and CPU usage and memory usage for cluster workload. Next, the workload is presented in the form of a time series $X = (x_1, x_2, ..., x_t)$, which is an arrangement of workload values in chronological order with fixed time intervals, and the workload value at time $t$ is $x_t$. In this case, the time intervals are 60min and 5min for web and cluster workloads, respectively.

**SG filter:**

The resulting workload time series data from the previous part are likely to be contaminated by noise caused by some abnormal events, such as machine failures [43]. Therefore, they must be cleaned before applying the prediction model. In this paper, we use the SG filter [14] to clean data from possible outliers and noise.

The SG filter is a smoothing method that uses convolution and polynomial least squares approximation to remove noise while preserving the width and the peak of the original signal. According to [43], this filter outperforms the average filter and the median filter in terms of forecasting accuracy. Equation 1 yields the polynomial approximation, with $2M + 1$ samples as inputs ($c = 0$ is considered to be the center):

$$\text{poly}(c) = \sum_{i=0}^{\text{pow}} \alpha_i c^i$$

where $\text{pow} \leq 2M + 1$ is the power of the polynomial.

Next, Equation 2 describes the error between the estimated and raw values and yields the polynomial solution:

$$\epsilon_{\text{pow}} = \sum_{c=-M}^{M} (\text{poly}(c) - x(c))^2$$

Finally, the outputs are calculated as follows:

$$\text{poly}(c) = \sum_{c=-M}^{M} h(j)x(c-j)$$

**Min-Max Normalization:**

After cleaning the data with an SG filter, they go through a normalization process. We use the min-max normalization to scale the workload time series in the same range of [0,1]. This normalization is described by the following formula:

$$X_{\text{norm}} = (X - X_{\text{min}})/(X_{\text{max}} - X_{\text{min}})$$

, where $X$ and $X_{\text{norm}}$ are the time series before and after normalization, respectively; and $X_{\text{min}}$ and $X_{\text{max}}$ are the minimum and the maximum of the time series $X$, respectively.

2) Chaotic Analysis of Cloud Workload Time Series

The term chaos theory is somewhat confusing: many researchers use terms such as nonlinear dynamics, self-organizing theory, bifurcation theory, or change theory [44]. Even though chaos theory consists of elements of chance, chaos is not random disorder. Instead, chaos is an action between certainty and randomness, and is thus uncertain in the long term but feasible to estimate in the short term.

To confirm the chaotic nature of cloud workload time series and investigate its chaotic features, it is possible to perform a chaotic time series analysis over the cloud workload time series. The chaoticity of a time series can be inspected by its corresponding maximal Lyapunov exponent (MLE) [45]. To calculate the MLE, the time series must first be embedded in a multidimensional space called phase space to form the trajectory matrix. In such an embedding, every point in multidimensional space is a vector whose elements are the delayed version of the time series. A detailed description of the methods used for the reconstruction of the phase space, and the calculation of the MLE is described below.

**Reconstruction of the phase space:**

The state of a system can be described by its state variables. The state variables at time $t$ form a vector in an $m$-dimensional space, which is called phase space. The state of

![Figure 3: Overview of the proposed workload prediction method](image-url)
a system typically changes over time, and thus, the vector in the phase space describes a trajectory representing the time evolution or the dynamics of the system. The observation of a real process usually does not yield all possible state variables. Either not all state variables are known or not all of them can be measured. However, because the evolution of any state variable in a system is dependent on other state variables that interact with it, it is possible to reconstruct the phase space using one observation state variable of the system.

According to Takens’ delay embedding theorem [46], the time series of any variable of the system can be used to reconstruct the phase space; the phase space reconstruction achieved by embedding a single-variable time series is not exactly the same as the original phase space, but its topological properties are preserved. For example, we let \( \{ x_i : i = 1, 2, ..., N \} \) be a single variable time series of a cloud workload, where \( N \) is the length of the workload time series. From this time series, an \( m \)-dimensional phase space can be reconstructed, and an embedded phase vector is then described by:

\[
X_i = (x_{i-(m-1)\tau}, x_{i-(m-2)\tau}, ..., x_{i})
\]

where \( \tau \) is the time delay; \( m \) is the embedding dimension; and \( X_i \) is a phase-space vector at discrete time \( i \), where \( i \in [1 + (m-1)\tau, N] \).

To guarantee an accurate reconstruction of phase space that can restore attractors in phase space, optimal values of time delay \( \tau \) and embedding dimension \( m \) should be determined. Studies have reported that the suitable value of \( \tau \) should be sufficiently large that \( x_i \) and \( x_{i+\tau} \) are rather independent but not so large that they are completely independent in a statistical sense [47]. Similarly, if \( m \) is too large, the density of points defining the attractor will decrease, and the level of contamination of the data will unnecessarily increase due to the noise in the data [48]. However, if \( m \) is too small, distant points on the attractor will overlap, and the attractor will be folded.

To estimate the optimal value of \( \tau \), we use the average mutual information method (AMI) [49]. The AMI approach measures the nonlinear dependence between successive points \( x_i \) and \( x_{i+\tau} \) based on Shannon’s entropy; this is the information we already know about the value of \( x_{i+\tau} \) if we know \( x_i \). The AMI function is defined as:

\[
I(\tau) = \sum_{x_i, x_{i+\tau}} P(x_i, x_{i+\tau}) \log_2 \left( \frac{P(x_i, x_{i+\tau})}{P(x_i)P(x_{i+\tau})} \right)
\]

where \( P(x_i, x_{i+\tau}) \) is the joint probability density in values of \( x_i \) and \( x_{i+\tau} \), \( P(x_i) \), and \( P(x_{i+\tau}) \) are the individual probability densities in values of \( x_i \) and \( x_{i+\tau} \), respectively. When the time delay becomes large, the chaotic behavior of the signal makes the measurements \( x_i \) and \( x_{i+\tau} \) become independent in a practical sense, and \( I(\tau) \) will tend to zero. According to [49], the suitable value of the time delay corresponds to the time delay when the AMI reaches its first minimum.

To estimate the embedding dimension \( m \), we use the False Nearest Neighbors (FNN) method introduced by Kennel et al. [50]. This method searches for the nearest neighbors of every point in a given dimension \( m \) and then checks to see if these points are still near neighbors in one higher dimension \( m+1 \). If this condition is verified, then the dimension \( m \) is the correct embedding dimension, but if it is not verified, then the points that were considered close neighbors in the dimension \( m \) are actually “false neighbors”, and the dimension \( m \) is not the correct embedding dimension. This process is repeated at higher dimensions until the percentage of FNN becomes zero or sufficiently small.

**Calculation of MLE:**

The calculation of the MLE is the next step of the chaotic time series analysis that follows the reconstruction of the phase space. The MLE is used to inspect the sensitive dependence on the initial conditions (i.e., the presence of chaotic attractors).

Each dynamic system is characterized by the entire spectrum of Lyapunov exponents, \( \lambda_k(k = 1, 2, ..., m) \), where \( m \) is the embedding dimension. Lyapunov exponents play a fundamental role in the description of the behavior of dynamical systems because they describe how orbits on the attractor move apart (or together) under the evolution of the dynamics [47]. Lyapunov exponents show the exponential convergence or divergence of nearby points in the multidimensional space, and MLE is the maximum Lyapunov exponent that determines the largest convergence or divergence. In most cases, checking the existence of chaos requires only a calculation of the MLE of the system. A positive value of the latter indicates chaos, while a negative value indicates no chaos.

Various estimation methods of MLE are described in the literature [48], [51]. In this paper, we use the algorithm proposed by Rosenstein et al. [52]. The first step of this algorithm consists of reconstructing the phase space from a single time series using the method described in Section III.B.2. After reconstructing the phase space, the algorithm looks for the nearest neighbor of each point on the trajectory. The nearest neighbor of the reference point \( X_i \) is assumed to be \( \hat{X}_i \), which satisfies the following:

\[
d_i(0) = \min_{X_i} \| X_i - \hat{X}_i \|
\]

where \( d_i(0) \) is the initial distance from \( X_i \) to its nearest neighbor \( \hat{X}_i \). In addition to this constraint, nearest neighbors should have a temporal separation greater than the mean period of the time series:

\[
|i - \hat{i}| > \text{meanperiod}
\]
\[ d_i(j) \approx C_i e^{\lambda_1(j\Delta t)} \]  

where \( d_i(j) \) is the distance between the \( i \)th pair of nearest neighbors after \( j \) discrete-time steps, i.e., \( j\Delta t \) seconds; \( C_i \) is the initial separation; and \( \lambda_1 \) is the MLE. By taking the logarithm of both sides of Equation 9, we find:

\[ \ln d_i(j) \approx \ln C_i + \lambda_1(j\Delta t) \]  

Equation 10 corresponds to a set of approximately parallel lines, each with a slope roughly proportional to \( \lambda_1 \). Therefore, the MLE can be simply calculated using the least-squares fit to the “average” line defined by:

\[ y_j = \frac{1}{\Delta t}(\ln d_i(j)) \]

where \((.\) is the value averaged for all \( j \).

3) CANFIS model building

As mentioned earlier, it is possible for a chaotic system to be predicted in the short term. Therefore, in this section, we present the proposed CANFIS model that we will use to perform workload prediction.

The ANFIS model [10] is a combination of two intelligence systems, a neural network (NN) system and a fuzzy inference system (FIS), in which the NN learning algorithm is used to determine the parameters of the FIS. NNs are nonlinear statistical data modeling tools that can capture and model any input-output relationships (or can learn to detect complex patterns in data). FIS is the process of formulating the mapping from a given input to an output using fuzzy logic. This mapping provides a basis from which decisions can be made or patterns can be discerned.

The contributions of this paper regarding the ANFIS model concern its starting point, which is the selection of the inputs. These inputs are selected using the phase space reconstruction method of chaos theory. The proposed chaos ANFIS model is referred to in the remainder of this paper as the CANFIS model.

As mentioned in Section III.B.2, the embedding dimension corresponds to the lowest dimension that can describe the attractor’s motion. As mentioned before, the cloud workload system condition can be recovered from one observed workload time series. Thus, if we select the embedding dimension of the collected workload data (e.g., CPU usage, memory usage, number of requests) as the dimension of the input space of the ANFIS model, we can restore the dynamical features of the workload system in the embedding phase space. Therefore, the number of ANFIS input vectors will be equal to or higher than the embedding dimension. This methodology resolves the problems of ANFIS inputs selection and considers the nonlinear features of the cloud workload system. Thus, as presented in Equation 5, the input \( X \) and the output \( Y \) of the CANFIS prediction model are:

\[ X = \begin{bmatrix} X_1 + (m-1)\tau \\ X_2 + (m-1)\tau \\ \vdots \\ X_N - (m-1)\tau \end{bmatrix} \]

\[ Y = \begin{bmatrix} X_{i+1} + (m-1)\tau + 1 \\ \vdots \\ X_{N+1} \end{bmatrix} \]

The next step in the process of CANFIS model construction is the selection of the type and the number of membership functions (MFs) involved in the FIS process. The selection can be performed using expert knowledge, empirically (i.e., by examining the desired input-output data), and/or by trial and error if no expert is available [10]. The rule base used in this work contains if-then rules of Sugeno type described as follows:

Rule \( j \): If \( X_1 + (m-1)\tau \) is \( A_1^{i+1} \) and \( X_2 + (m-1)\tau \) is \( A_2^{i+1} \) and \( \ldots \) and \( X_N \) is \( A_N^{i+1} \), then:

\[ f_j = z_1^j X_1 + z_2^j X_2 + \ldots + z_N^j X_N + u^j \]

where \( j = 1, 2, \ldots, n_c \) is the number of rules, \( \{ X_i, i = 1 + (m - 1)\tau, \ldots, N \} \) is the input vector, \( A_i^{j} \) is the linguistic label associated with the \( i \)th input variable in the \( j \)th fuzzy rule, \( f_j \) is the prediction result according to the \( j \)th fuzzy rule, and \( z_k^j \) and \( u^j \) are the parameters that are determined during the training process.

The CANFIS architecture that implements these rules is shown in Figure 4 and consists of five layers: the fuzzy layer, the product layer, normalized layer, defuzzy layer and total output layer. The node function in each layer is presented below. Note that fixed nodes, which are represented by circles, refer to the parameter sets that are fixed in the system, while adaptive nodes, represented by squares, refer to the parameter sets that are adjustable in these nodes.

![Figure 4: CANFIS architecture](image-url)
TABLE 3: Two passes in the hybrid learning algorithm for CANFIS.

| PP    | CP   | Signals | Node Outputs | Error Rates |
|-------|------|---------|--------------|-------------|
| Fixed | LSE  |         |              |             |

Layer 1 (Fuzzy layer): All the nodes in this layer are adaptive nodes with node functions described as:

\[ O_1^j = \mu_{A_1^j}(X_i) \]  

where \( i = 1 + (m - 1)r, ..., N, j = 1, 2, ..., n_r \) and \( O_1^j \) is the output of layer 1 with respect to the \( i \)th input \( X_i \) in the \( j \)th fuzzy rule, and \( \mu_{A_1^j}(X_i) \) is the MF of \( A_1^j \), which specifies the degree to which the given \( X_i \) specifies the quantifier \( A_1^j \).

Any continuous and piecewise differentiable functions, such as generalized bell-shaped MFs, triangular-shaped MFs, or Gaussian MFs, can be used in this layer. For example, if the Gaussian MF is used, it is expressed as:

\[ \mu_{A_1^j}(X_i) = \exp \left[ -\frac{(X_i - c_j)^2}{2\sigma_j^2} \right] \]  

where \( i = 1 + (m - 1)r, ..., N, j = 1, 2, ..., n_r \) and \{\( c_j, \sigma_j \)\} is the parameter set. The parameters in this layer are referred to as the premise parameters (PPs).

Layer 2 (product layer): Each node in this second layer is a fixed node, labeled \( \prod \), with the node function to be multiplied by input signals to serve as output:

\[ w^j = \prod_{i=1+(m-1)r}^{N} \mu_{A_1^j}(X_i), j = 1, 2, ..., n_r \]  

where \( w^j \) is the firing strength of the \( j \)th rule.

Layer 3 (Normalized layer): Each node in this layer is also a fixed node that is labeled \( N \) with the node function to normalize \( w^j \), which calculates the ratio of the \( j \)th rule’s firing strength to the sum of all rules’ firing strengths. Each node is defined as follows:

\[ \bar{w}^j = \frac{w^j}{\sum_{j=1}^{n_r} w^j}, j = 1, 2, ..., n_r \]  

where \( \bar{w}^j \) refers to the normalized firing strength.

Layer 4 (Defuzzy layer): Each node in this layer is an adaptive node with a node function:

\[ O^j = \bar{w}^j f^j \]  

\[ = \bar{w}^j(z_i^{(1+(m-1)r)}X_i^{(1+(m-1)r)} + ... + z_N^{(n_r)}X_N + w^j) \]  

where \( j = 1, 2, ..., n_r \) and \{\( z_i^j, w^j \)\} is the parameter set related to the first-order polynomial. Parameters in this layer are referred to as the consequent parameters (CPs).

Layer 5 (Total output layer): The single node in this layer is a fixed node labeled \( \Sigma \) with a node function to compute the overall output as the summation of all incoming signals, and can be expressed as:

\[ O^5 = \text{overall output} = Y = \sum_{j=1}^{n_r} \bar{w}^j f^j \]

\[ = \sum_{j=1}^{n_r} \bar{w}^j \sum_{i=1}^{n_r} w^j \]  

The proposed CANFIS model uses a hybrid learning algorithm that combines least squares estimation (LSE) with gradient descent (GD). Every epoch of this algorithm is composed of a forward pass (FP) and a backward pass (BP). In the FP, the PPs are fixed. The functional signals go forward until layer 4, and the CPs are identified by the LSE. Once the optimal CPs are calculated, the backward pass begins. In this pass, the GD is used to adjust the PPs. The output of the CANFIS is calculated by setting the CPs to the values found in the FP. By comparing the estimated output with the real output, the output error rates of the CANFIS propagate backward from the output to the input to adjust the PPs using the standard backpropagation algorithm. Table 3 summarizes the activities in each pass.

The training of the CANFIS model continues until the specified number of epochs or the specified root mean square error (RMSE) is achieved. When training is complete, the training error and final MFs from the training data set are generated. Although it is feasible for CANFIS to perform with only a training data set, it is preferable to use a checking data set too to enhance the accuracy and avoid overfitting the training data. If the resulting checking error does not meet the target error, the entire process of selecting CANFIS parameters (i.e., type of MFs, number of MFs, epochs) should be repeated until the goal error is met. Then, the testing data set is used to evaluate the performance of the CANFIS model.

C. TIME COMPLEXITY

This section describes the time complexity of the proposed prediction method. The first step is data cleaning using SG filter. For a given time series of length \( N \), the maximum amount of time required to execute this step is \( O(N) \). The second step is the estimation of the time delay \( \tau \) using AMI method and requires \( O(N \log(N)) \). The third step is the calculation of the embedding dimension \( m \) using the FNN method and requires \( O(N^2) \). Next, the phase space reconstruction takes \( O(N) \). Also, the calculation of MLE takes \( O(N^2) \). The last step is the building of the CANFIS model. Considering the MF type to be Gaussian, this step takes \( O(N^2) \). Finally, the total time complexity of the proposed method is \( O(N \log(N)) + O(N) + O(N^2) \).

IV. PERFORMANCE EVALUATION

This section presents the workload datasets used to evaluate the proposed prediction method, the data preprocessing stages, the experimental setup, the evaluation metrics, the
baseline methods used to evaluate the performance of the proposed model and the results of the performed experiments.

A. DATASETS

To evaluate the prediction method introduced in this paper, we used two types of cloud workload traces from real world applications: 1) web workload traces from Wikipedia servers [53] and NASA servers [54]; and 2) cluster workload traces from Google cluster servers [55]. These two types of workloads have different features that allow us to evaluate the proposed proposal in a more reliable, generic and realistic way. A summary of the evaluated workload information is given in Table 4, and detailed information on these workloads is presented in the following paragraphs.

**Web Workload Traces:** The web workloads describe the behavior of web application models on cloud computing. In this paper, we use two web workload traces: the Wikipedia trace [53] and the NASA trace [54].

The Wikipedia trace is used in many different studies [16], [37] and contains the number of http requests that are received for each project resource, such as static pages and images, which were collected in 1-hour intervals. The trace also contains the project name and the language of each accessed resource. In this experiment, we consider only requests to English Wikipedia resources dated from midnight, January 01, 2015 to 23 pm, March 31, 2015 (a total of three months). The Wikipedia workload is shown in Figure 5(a). As shown in the figure, the Wikipedia time series exhibits strong seasonal behaviors in addition to some trends.

The NASA trace is also used in many different studies [4], [22] and consists of two logs that contain the HTTP requests to NASA Kennedy Space Center WWW server in Florida. The first log was collected between 00:00:00 July 1, 1995 and 23:59:59 July 31, 1995. The second log was

![Figure 5: Original web workload traces with 60-minute time intervals. (a) Wikipedia trace (b) NASA trace.](image)

![Figure 6: Original cluster workload traces with 5-minute time intervals. (a) Google CPU usage trace (b) Google memory usage trace.](image)

**TABLE 4: Summary of the evaluated web and cluster workloads.**

| Workload | Trace          | Duration | Prediction Interval |
|----------|----------------|----------|---------------------|
| Web      | Wikipedia      | 90 days  | 60 min              |
|          | Nasa           | 60 days  |                     |
| Cluster  | Google CPU     | 29 days  | 5 min               |
|          | Google Memory  | 29 days  |                     |
collected between 00:00:00 August 1, 1995 and 23:59:59 August 31, 1995. During these two months, the server received 3,461,612 requests with timestamps with a 1-second resolution. We note that from 01/Aug/1995:14:52:01 until 03/Aug/1995:04:36:13, there was no access recorded because the web server was shut down due to the Hurricane Erin [54]. This workload is shown in Figure 5(b).

The NASA dataset exhibits repetition in some periods. However, compared to the Wikipedia workload, the NASA workload is more dynamic, which means that the requests issued to the NASA website might increase by a larger margin and a faster speed.

**Cluster Workload Traces:** The cluster workloads describe the behavior of big data analytic and cluster application models on clouds. Similar to many other studies [43], [26], this paper considers two cluster workload traces (CPU and memory usages) of the Google cluster [55]. Google Workload is a 29-day workload trace published by Google in 2011 that contains approximately 650000 jobs, 20 million tasks and 12500 machines. The job includes one or more tasks, and every task may consist of multiple processes. The information related to each job and task, such as the job ID, the task ID, the measurement period, the mean CPU usage rate, the canonical memory, the mean local disk space used and others are given in this trace.

Figures 6(a) and 6(b) show the two traces and highlight that cluster workloads are dynamic and vary markedly compared to web workloads, which makes it challenging to predict this type of workload.

### B. EXPERIMENTAL SETUP

In this paper, we perform experiments on a machine equipped with an Intel® Core i5-6300U processor running at 2.40 GHz and 16 GB of memory. We use MATLAB R2017a software to implement the method and perform simulations. The statistical analysis was conducted using the STAC web platform [56]. The data traces are split into training, checking and testing parts. The first 60% of the data are used to train the model, the next 20% of data are used for checking, and the remaining 20% are used to test the model. All experiments are repeated 10 times, and the mean accuracy is stated to mitigate randomness. Values of $M$ and $pow$ for the SG filter are set equal to 6 and 3, respectively. Table 5 shows the primary parameters used in the building of the CANFIS model. The impact of these parameters on the prediction results will be discussed later in this paper.

### C. PERFORMANCE METRICS

The goals of the evaluation of the proposed prediction method are twofold. First, we measure the prediction accuracy for the evaluated workloads. Second, we measure the training time of the proposed prediction method because prediction in a predetermined time is essential for every workload prediction method.

To evaluate the prediction accuracy of the proposed method, we use the following metrics, which are widely used in many different studies [22], [34], [43], [57]. For all metrics, $x_i$ and $\hat{x}_i$ are the real and predicted values of the workload, respectively, and $n$ is the forecast horizon:

- **Root mean square error (RMSE):** This is a quadratic scoring rule that measures the errors’ average magnitude. Thus, RMSE can be described as the square root of the average of the squared differences between real and predicted values. The RMSE is beneficial when large errors are specifically unwanted because it gives high weights to large errors. A lower value of RMSE indicates higher accuracy in the prediction method. The formula of the RMSE metric is given in Equation 21:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \hat{x}_i)^2}{n}}$$  \hspace{1cm} (21)

- **Mean absolute percentage error (MAPE):** MAPE is the average of the absolute values of the differences between the real and predicted values. Unlike RMSE, MAE assigns equal weights to all errors because it is a linear score. A smaller MAE value indicates a better fit of the forecasting model. Equation 22 presents the mathematical formula of the MAE metric:

$$MAE = \frac{\sum_{i=1}^{n} |x_i - \hat{x}_i|}{n}$$  \hspace{1cm} (22)

- **Mean absolute error (MAE):** The MAE is the average of the absolute values of the differences between the real and predicted values. Unlike RMSE, MAE assigns equal weights to all errors because it is a linear score. A smaller MAE value indicates a better fit of the forecasting model. Equation 22 presents the mathematical formula of the MAE metric.

- **$R^2$ coefficient:** This coefficient measures how well the prediction model fits the real data. Its value lies in the range $[0,1]$, and the closer the value is to 1, the better the prediction model. The corresponding mathematical formula is given in Equation 24:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (x_i - \hat{x}_i)^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$  \hspace{1cm} (24)

where $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} (x_i)$. 

---

**TABLE 5:** Parameters of CANFIS model

|                  | Web workloads | Cluster workloads |
|------------------|---------------|-------------------|
|                  | Wikipedia     | Nasa              | CPU    | memory |
| Type of MFs      | gaussmf       | gaussmf           | gaussmf| gaussmf |
| Number of MFs    | 2             | 3                 | 2      | 3       |
| Epochs           | 10            | 80                | 10     | 10      |

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3174061, IEEE Access
D. BASELINES

To validate the effectiveness of investigating the chaoticity of cloud workloads and selecting ANFIS inputs based on chaos theory, we compare the proposed method with the baseline ANFIS model (called the ANFIS model in this paper). We also compare the CANFIS model against three well-known baseline models: the LSTM, SVM, and ARIMA models. The following paragraphs provide a description of these baselines.

LSTM: The long short-term memory (LSTM) network [58] is an advanced recurrent neural network that can memorize data sequences from earlier stages for the aim of future use. This process is possible due to the gates and the memory line that are incorporated in the LSTM network.

SVM: Support vector machine (SVM) [59] is a machine learning model that is used in both classification and regression problems. SVM operates by constructing a hyperplane in multidimensional space to separate the data into classes. Using an iterative method, SVM generates an optimal hyperplane with minimum error.

ARIMA: The autoregressive integrated moving average (ARIMA) model [16] is a statistical model that is used to predict time series. The model begins by transforming the nonstationary time series data into stationary data using a d-order difference method. Then, the autocorrelation and the partial autocorrelation functions of the resulting time series are plotted and analyzed to determine the suitable orders p and q for the autoregressive model’s lags and the moving average model’s lags, respectively. Finally, the model uses the least-squares method to estimate parameters and then makes predictions of the future values.

E. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we present the results of the different experiments performed using the selected workloads to validate the performance enhancement of the proposed method. First, the results of the chaotic analysis of cloud workload are presented. Then, the prediction results of the proposed method

TABLE 6: The values of $\tau$, $m$ and MLE for web and cluster workloads

| Workload | Trace     | $\tau$ | $m$  | MLE      |
|----------|-----------|--------|------|----------|
| Web      | Wikipedia | 7      | 5    | 2.25E-05 |
|          | Nasa      | 8      | 5    | 3.42E-05 |
| Cluster  | Google CPU| 1      | 3    | 0.0017   |
|          | Google Memory | 1   | 3    | 0.0019   |
are given, and an evaluation of the prediction accuracy of the proposed method is made by conducting a comparative study between the proposed method and the baselines and some other new neural network-based methods. Finally, a statistical analysis is performed to confirm the efficiency of the proposed model.

1) Results of Chaotic Analysis of Cloud workload
As mentioned in Section III.B.2, the first step in chaotic time series analysis is the reconstruction of the phase space by determining the optimal values of $\tau$ and $m$. Taking the Wikipedia and NASA workloads as examples, the AMI and the percentage of FNN of the time series are calculated and displayed in Figures 7 and 8.

As shown in the results, the AMI of the Wikipedia and NASA time series reaches its first minimum at the time delay equal to 7 and 8, respectively. In contrast, the percentage of FNN becomes zero when the embedding dimension is 5 for both Wikipedia and NASA time series. These time delay and embedding dimension values will be used for phase space reconstruction of the Wikipedia and NASA time series. The phase space reconstruction parameters for Google CPU and Google memory time series are calculated using the same methods and are presented in Table 6.

Now, the phase space can be reconstructed, and a further investigation of chaos characteristics in cloud workloads can be performed. This investigation can be done by calculating the MLE for each time series using the method described in Section III.B.2. The results of the MLE for every time series are also presented in Table 6. According to the results, the MLEs of different workload data are all positive, demonstrating that the workload time series exhibits chaotic behavior.

2) Prediction accuracy
Figure 9 shows the results of the prediction of the web workloads using the CANFIS model. The proposed CANFIS model clearly captures the Wikipedia workload and the NASA workload patterns effectively, and the forecasted values of the workloads using CANFIS are very near the real values. Therefore, the CANFIS model can achieve high prediction accuracy for both the Wikipedia and NASA workloads.

To demonstrate the effectiveness of the CANFIS model, a comparison with four baseline methods (i.e., ANFIS, LSTM, SVM, ARIMA) using the CANFIS model is shown in Figure 10. All RMSE results are normalized to the CANFIS RMSE results. The value 1.00 refers to the results of the proposed CANFIS model, while higher values indicate poor performance. The RMSE metric in the latter formula can be replaced with any other error metric. On average, in terms of RMSE, the CANFIS model reduces the prediction error up to 77.58%, 80.35%, 79.21% and 80.54% against ANFIS, LSTM, SVM and ARIMA, respectively, on web workloads. This result occurs because the CANFIS model can recognize the seasonality and trends within this type of workload and can concurrently follow their changes.

Table 7 shows the MAE, MAPE and $R^2$ outputs for the two web workloads. Similar to the RMSE results, the CANFIS model achieves higher accuracies in terms of these metrics compared to the baseline methods for both the Wikipedia and NASA workloads. For example, MAE for the CANFIS model with Wikipedia and NASA workloads are 0.0056 and 0.0055, respectively, meaning that on average, the CANFIS model reduces MAE up to 77.31%, 79.76%, 78.41% and 80.12% compared to ANFIS, LSTM, SVM and ARIMA, respectively. Similarly, on average, the CANFIS model can reduce the error in terms of MAPE up to 80.01%, 82.41%, 81.11% and 82.69% against ANFIS, LSTM, SVM and ARIMA, respectively. Regarding the $R^2$ coefficient, the values of this coefficient for the CANFIS model with Wikipedia and NASA workloads are 0.9990 and 0.9960, respectively, which are near 1, indicating that the proposed model has a good fit.

Figure 11 shows the cumulative distribution function (CDF) of forecasting errors in web workloads. The x-axis denotes the absolute prediction error (i.e., $|Actual_t − Prediction_t|$), while the y-axis represents the cumulative probability of the errors. The curves for the CANFIS model are skewed to the left more than the curves for the baseline methods, which means that the majority of the CANFIS forecasting errors are smaller than those of the baseline methods for both the Wikipedia and NASA workloads. In addition, the results from the baseline methods have longer tails, meaning that they produce more extreme forecasting errors.

The results of the prediction of the cluster workloads using the CANFIS model are shown in Figure 12. Again, similar to the web workloads, the predicted values of CPU usage and memory usage using the proposed CANFIS model are near the real values, indicating that the CANFIS model can achieve good prediction accuracy for cluster workloads. This result can be confirmed by looking at the RMSE results shown in Figure 13. On average, the CANFIS model can reduce the RMSE error up to 65.35%, 67.26%, 65.47% and 66.95% against ANFIS, LSTM, SVM and ARIMA, respectively, on cluster workloads.

By considering the other prediction errors shown in Table 8, the CANFIS model has the lowest errors for the two cluster traces. For example, on average, the CANFIS model shows a reduction of MAE error up to 62.74%, 65%, 63.08% and 63.92% against ANFIS, LSTM, SVM and ARIMA models, respectively. The CANFIS model shows MAPE results of 2.2888% and 1.5273% with CPU usage and memory usage, respectively. These results imply, on average, a 56.69%,
FIGURE 9: Actual vs predicted web workloads using the proposed method. (a) Wikipedia workload prediction results. (b) Enlarged part. (c) NASA workload prediction results. (d) Enlarged part.

FIGURE 10: Normalized RMSE results in web workloads

TABLE 7: Performance comparison of the proposed and the baseline methods with web workloads

| Method | Wikipedia | | | Nasa | | | 
|---|---|---|---|---|---|---|
| | MAE | MAPE | $R^2$ | MAE | MAPE | $R^2$ |
| ARIMA | 0.0331 | 9.4340 | 0.9612 | 0.0241 | 16.6462 | 0.8636 |
| SVM | 0.0291 | 8.3371 | 0.9683 | 0.0230 | 15.9059 | 0.8710 |
| LSTM | 0.0336 | 9.1933 | 0.9605 | 0.0231 | 16.5657 | 0.8735 |
| ANFIS | 0.0261 | 7.4001 | 0.9751 | 0.0230 | 16.2764 | 0.8769 |
| CANFIS | 0.0056 | 1.7071 | 0.9990 | 0.0055 | 2.7497 | 0.9960 |
FIGURE 11: Cumulative distribution function (CDF) of prediction errors of the proposed and baseline methods for web workloads. (a) CDF for Wikipedia Trace. (b) CDF for NASA Trace.

FIGURE 12: Actual vs predicted cluster workloads using the proposed method. (a) Google CPU prediction results. (b) Enlarged part. (c) Google memory prediction results. (d) Enlarged part.

TABLE 8: Performance comparison of the proposed and the baseline methods with Cluster workloads

|            | CPU        | Memory     |
|------------|------------|------------|
|            | MAE        | MAPE       | R^2   | MAE        | MAPE       | R^2   |
| ARIMA      | 0.0285     | 8.6335     | 0.9200| 0.0177     | 2.7399     | 0.8313|
| SVM        | 0.0295     | 8.6985     | 0.9130| 0.0167     | 2.5556     | 0.8603|
| LSTM       | 0.0290     | 8.7924     | 0.9186| 0.0184     | 2.8271     | 0.8284|
| ANFIS      | 0.0285     | 8.4700     | 0.9171| 0.0168     | 2.5631     | 0.8579|
| CANFIS     | 0.0080     | 2.2888     | 0.9973| 0.0078     | 1.5273     | 0.9938|
59.96%, 56.95% and 58.86% reduction in the MAPE error compared to ANFIS, LSTM, SVM and ARIMA, respectively. The values of the $R^2$ coefficient for the CANFIS model with CPU usage trace and memory usage trace are 0.9973 and 0.9938, respectively. Because these values are near 1, we conclude that although cluster workloads do not contain a stable seasonality and trend, the CANFIS model can still produce predictions with higher accuracy.

The CDF of forecasting errors in cluster workloads is shown in Figure 14. Both the CPU usage and memory usage CDF curves for the CANFIS model are skewed to the left more than the curves for the baseline methods. Again, this result indicates that most prediction errors for both CPU usage and memory usage are small compared to those of the baseline methods.

In addition to the baseline methods, the proposed approach is also compared with some new neural network-based methods, EQNN [28] and SDWF [26]. Table 9 shows the error reduction percentage $\text{Error}_{\text{reduction}}$ of the proposed model against the ARIMA model and the $\text{Error}_{\text{reduction}}$ of the EQNN [28] and SDWF [26] models against the ARIMA model. This percentage is calculated using Equation 25 presented earlier in this section. In this study, the $\text{RMSE}_B$ corresponds to the RMSE of the ARIMA model, while the $\text{RMSE}_P$ corresponds to the RMSE of the proposed CANFIS model, EQNN model [28], or the SDWF model [26].

|                | Nasa | Google CPU | Google Memory |
|----------------|------|------------|---------------|
| EQNN [28]      | 54.30% | 19.82%    | 64.84%        |
| SDWF [26]      | -56.67% | -7.21%    | 53.13%        |
| CANFIS         | 77.72% | 74.68%    | 59.18%        |

The error reduction percentage of the proposed CANFIS model against the ARIMA model is much higher than the error reduction percentage of EQNN [28] and SDWF [26] models against the ARIMA model with both the NASA and Google CPU datasets. For the Google memory dataset, the performance of the three evaluated models is similar with SDWF [26] being the model with the lowest error reduction percentage (i.e., 53.13%), followed by the proposed (i.e., 59.18%) and the EQNN [28] model (i.e., 64.84%). Therefore, the proposed CANFIS model outperforms the existing forecasting models and can thus improve workload prediction accuracy.
3) Impact of model parameters on prediction results

The performance of the CANFIS model depends on the values of its initial parameters. Therefore, in this section, we evaluate the impact of some of these parameters, such as the type of MFs, the number of MFs, and the number of epochs, on the accuracy of the prediction results in terms of RMSE for every dataset used.

Impact of the type and the number of MFs:

To build the CANFIS model, various types of MFs (i.e., Gaussian, Generalized Bell, triangular, etc.) are used, and the number of MFs is varied from 1 to 3 for each input data. Tables 10-13 show the impact of the number and the type of MFs on the prediction accuracy for every workload dataset in terms of RMSE with epoch=10. The best CANFIS model that produces the smallest RMSE error is the model with 2 Gaussian MFs for the Wikipedia workload and 3 Gaussian MFs for the NASA workload. For the Google CPU workload, the CANFIS model with 2 MFs of three different types: Gaussian, Generalized Bell, and triangular, yields the same smallest value of RMSE (i.e., 0.0115). For the Google memory workload, the most suitable CANFIS model is the one with 3 Gaussian MFs.

Impact of the number of training epochs:

To evaluate the impact of the number of training epochs on the prediction results, we use different numbers of training epochs (i.e., 10, 50, 80, 120) to train the CANFIS model for every dataset. The results are shown in Table 14. We set the type and the number of MFs for every dataset based on the results of Tables 10-13. Increasing the number of epochs does not yield a marked improvement in the prediction accuracy. For the Wikipedia, Google CPU and memory workloads, using 10 epochs to train the CANFIS model is sufficient to give the smallest RMSE error. For the NASA workload, the smallest RMSE error is obtained for 80 epochs.

4) Training time

A comparison of the training time of the proposed method and some baseline methods with both web and cluster workloads is shown in Table 15. The LSTM model has the highest training time with all workloads. The results for the proposed

| Epochs | Wikipedia | Nasa | Google CPU | Google memory |
|--------|-----------|------|------------|---------------|
| 10     | 0.0073    | 0.0073 | 0.0115     | 0.0119        |
| 50     | 0.0074    | 0.0073 | 0.0115     | 0.0126        |
| 80     | 0.0074    | 0.0072 | 0.0115     | 0.0126        |
| 120    | 0.0075    | 0.0072 | 0.0115     | 0.0125        |

| Training time (s) | Web workloads | Cluster workloads | Average |
|-------------------|---------------|-------------------|--------|
|                   | Wikipedia     | Nasa              | CPU    | Memory |        |
| ARIMA             | 3.07          | 2.09              | 2.47   | 3.10   | 2.68   |
| SVM               | 0.44          | 0.38              | 1.43   | 1.29   | 0.88   |
| LSTM              | 114.60        | 76.59             | 630.13 | 616.84 | 359.54 |
| ANFIS             | 0.80          | 1.26              | 0.37   | 4.26   | 1.67   |
| CANFIS            | 0.76          | 2.92              | 0.53   | 1.68   | 1.47   |
and the other baselines vary depending on the workloads. For the Wikipedia workload, SVM has the lowest training time (0.44 s), followed by the proposed model (0.76 s), the ANFIS model (0.80 s) and the ARIMA model (3.07 s). For the NASA workload, the SVM again has the lowest training time (0.38 s), followed by the ANFIS model (1.26 s), the ARIMA model (2.09 s) and the proposed method (2.92 s). Regarding Google CPU usage, the ANFIS model has the lowest training time (0.37 s), followed by the proposed method (0.53 s), the SVM model (1.43 s) and the ARIMA model (2.47 s). For Google memory usage, SVM has the lowest training time (1.29 s), followed by the proposed method (1.68 s), the ARIMA model (3.10 s) and the ANFIS model (4.26 s). On average, the SVM model has the lowest training time among all the methods, followed by the proposed method, the ANFIS model, the ARIMA model and the LSTM model. Although the SVM model has less training time on average than the proposed model, it has less accuracy than the proposed CANFIS model for all workloads and less accuracy than the ANFIS model for the majority of the workloads. In addition, the average training time (1.47 s) of the proposed CANFIS model is still not important compared to the prediction interval and VM reconfiguration interval (e.g., VM startup time is more than 30 s), meaning that it will not have an impact on the resource provisioning process. Finally, there is always a possibility to reduce the training time of the proposed method by implementing it in parallel on high computing infrastructure.

F. STATISTICAL ANALYSIS

In this section, we validate the prediction accuracy of the proposed prediction model and some baselines using statistical analysis. The statistical analysis can determine whether the observed differences in experimental results are significant or simply occur due to random chance. In this context, we use the Friedman test along with Finner post hoc analysis [60], [61]. This test is widely used in many studies [28], [62] and is considered powerful when the performance of five or more models is compared [63]. The Friedman test assumes a null hypothesis (H₀ РФ ) that states that the primary results of the different algorithms are equal. The alternative hypothesis (H₁ РФ ) advocates that the mean of at least one algorithm’s results is different from others. The Friedman test also ranks the algorithms according to their performance.

The statistics of the Friedman test (statistics and p-values) for RMSE-, MAE-, and MAPE-based prediction accuracy evaluation are shown in Table 16. All p-values indicate a rejection of H₀ РФ, with a significance level of 0.05.

The Friedman ranks of the algorithms are shown in Table 17. The proposed method has the best rank among all five algorithms, meaning that the proposed method’s behavior has a significant difference.

Finner post hoc analysis is used to compare the five algorithms in pairs, considers the proposed method as a control method, and works around a null hypothesis (Hᵢ РФ) that assumes that the mean of the proposed method is equal to each other member of the group under test. Table 18 shows the results of the pairwise comparison obtained using the Finner post hoc analysis. The Hᵢ РФ is rejected for the proposed method against the SVM, LSTM, and ARIMA models for RMSE-based prediction accuracy evaluation and against LSTM and ARIMA for MAE- and MAPE-based prediction accuracy evaluation. Conversely, H₀ РФ is accepted for the proposal against ANFIS for RMSE-based accuracy evaluation and against ANFIS and SVM for MAE- and MAPE-based accuracy evaluation. However, this hypothesis will be rejected against these methods at a significance level of 0.2 or even 0.1 against SVM for MAE-based accuracy evaluation. Based on the encouraging results of the proposed model and their statistical analysis, the superior performance of the proposed model can be confirmed.

V. CONCLUSION

In this study, we investigated the presence of chaos in cloud workloads. We used chaotic time series analysis to identify the chaotic behavior of the workloads and proposed an improved ANFIS model called the CANFIS model to predict future workloads. The inputs of the proposed model are selected using the phase space reconstruction method and embedding theory.

Using real-world web and cluster workloads, we performed experiments to evaluate the performance of the proposed model. Every workload used in these experiments went through several preprocessing operations before being fed to the proposed model. These operations include extraction, aggregation, cleaning, and normalization. Cleaning was performed using the SG filter, which can effectively eliminate noise and outliers without changing the width and the peak of the original signal.

The results of the experiments showed that the proposed CANFIS model achieved higher prediction accuracy than the ANFIS, LSTM, SVM and ARIMA models and some other new neural network-based methods. In addition, the proposed model has a low training time, meaning that it will not have any effect on the provisioning process.
TABLE 18: Post-hoc analysis using Finner test

| Comparison  | RMSE p-value | MAE p-value | MAPE p-value |
|-------------|--------------|-------------|--------------|
| CANFIS vs LSTM | 2.68328 0.02884 | H0 rejected | 3.13050 0.00696 | H0 rejected | 1.56525 0.17971 | H0 accepted | 3.14614 0.17971 | H0 accepted |
| CANFIS vs ARIMA | 2.49677 0.02884 | H0 rejected | 2.57148 0.02015 | H0 rejected | 1.90066 0.07572 | H0 accepted | 1.56525 0.15355 | H0 accepted |
| CANFIS vs SVM  | 2.23607 0.03365 | H0 rejected | 1.90066 0.07572 | H0 accepted | 1.56525 0.15355 | H0 accepted | 1.56525 0.15355 | H0 accepted |
| CANFIS vs ANFIS | 1.56525 0.11752 | H0 accepted | 1.34164 0.17971 | H0 accepted | 1.34164 0.17971 | H0 accepted | 1.34164 0.17971 | H0 accepted |

REFERENCES

[1] R. Buyya, C.S. Yeoh, S. Venugopal, J. Broberg, and I. Brandic, “Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility,” Future Generation Computer Systems, vol. 25, no. 6, pp. 599-616, 2009.

[2] S.K. Garg, A.N. Tousi, S.K. Gopalaiyengar, and R. Buyya, “SLA-based virtual machine management for heterogeneous workloads in a cloud datacenter,” Journal of Network and Computer Applications, vol. 45, pp. 108-120, 2014.

[3] R.L. Sri, and N. Balaji, “An empirical model of adaptive cloud resource provisioning with speculation,” Soft Computing, vol. 23, pp. 10983-10999, 2019.

[4] J. Kumar, D. Saxena, A.K. Singh, and A. Mohan, “BiPhase adaptive learning-based neural network model for cloud datacenter workload forecasting,” Soft Computing, vol. 24, pp. 14593-14610, 2020.

[5] M. Amiri, L.M. Khanli, and R. Mirandola, “An online learning model based on episode mining for workload prediction in cloud,” Future Generation Computer Systems, vol. 87, pp. 83-101, 2018.

[6] A. Ali-Eldin, O. Seleznjev, S. Sjostedt-de Luna, J. Tordsson, and E. Elmroth, “Measuring cloud workload burstiness,” in IEEE/ACM 7th International Conference on Utility and Cloud Computing, London, UK, 2014, pp. 566-572.

[7] A. Ali-Eldin, J. Tordsson, E. Elmroth, and M. Kihl, “Workload Classification for Efficient Auto-scaling of Cloud Resources,” 2005. Accessed on: Feb. 18, 2019. [Online]. Available: www8.cs.unu.na/research/umin/reports/2013/013/part1.pdf.

[8] R.C. Hibborn, Chaos and Nonlinear Dynamics: An Introduction for Scientists and Engineers, New York, USA: Oxford University Press, 1994.

[9] J. Wang, and Q. Shi, “Short-term traffic speed forecasting hybrid model based on chaos-wavelet analysis-support vector machine theory,” Transportation Research Part C: Emerging Technologies, vol. 27, pp. 219-232, 2013.

[10] J.S.R. Jang, “Anfis: Adaptive-network-based fuzzy inference systems,” IEEE Transactions on Systems, Man, and Cybernetics, vol. 23, no. 3, pp. 665-685, 1993.

[11] Z. Chen, Y. Zhu, Y. Di, and S. Feng, “Self-adaptive prediction of cloud resource demands using ensemble model and subtractive-fuzzy clustering based fuzzy neural network,” Computational Intelligence and Neuroscience, vol. 2015, pp. 1-14, 2015.

[12] X. Wu, H. Wang, D. Wei, and M. Shi, “ANFIS with natural language processing and gray relational analysis based cloud computing framework for real time energy efficient resource allocation,” Computer Communications, vol. 150, pp. 122-130, 2020.

[13] T.L. Anh, “Workload Prediction for Resource Management in Data Centers,” 2016. Accessed on: Jan. 1, 2019. [Online]. Available: https://umu.diva-portal.org/smash/get/diva2:957163/FULLTEXT01.pdf.

[14] A. Savitzky, and M.J.E. Golay, “Smoothing and differentiation of data by simplified least squares procedures,” Analytical Chemistry, vol. 36, no. 8, pp. 1627-1639, 1964.

[15] M. Amiri, and L. Mohammad-Khanli, “Survey on prediction models of resources provisioning in cloud,” Journal of Network and Computer Applications, vol. 82, pp. 95-113, 2017.

[16] R.N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, “Workload prediction using ARIMA model and its impact on cloud applications QoS,” IEEE Transactions on Cloud Computing, vol. 3, no. 4, pp. 449-458, 2015.

[17] Y.G. Tran, V. Debusschere, and S. Bacha, “Hourly server workload forecasting up to 168 h ahead using Seasonal ARIMA model,” in IEEE International Conference on Industrial Technology, Athens, Greece, 2012, pp. 1127-1131.

[18] N. Roy, A. Dubey, and A. Gokhale, “Efficient autoscaling in the cloud using predictive models for workload forecasting,” in IEEE 4th International Conference on Cloud Computing (CLOUD), Washington, DC, USA, 2011, pp. 500-507.

[19] Z. Amekraz, and M.Y. Hadi, “An adaptive workload prediction strategy for non-gaussian cloud service using ARMA model with higher order statistics,” in IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, 2018, pp. 646-651.

[20] E. Caron, F. Desprez, and A. Muresan, “Forecasting for grid and cloud computing on-demand resources based on pattern matching,” in IEEE 2nd International Conference on Cloud Computing and Technology and Science (CloudCom), Indianapolis, IN, USA, 2010, pp.456-463.

[21] Y. Hu, B. Deng, F. Peng, and D. Wang, “Workload prediction for cloud computing elasticity mechanism,” in IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), Chengdu, China, 2016, pp. 244-249.

[22] J. Kumar, and A.K. Singh, “Workload prediction in cloud using artificial neural network and adaptive differential evolution,” Future Generation Computer Systems, vol. 81, pp. 41-52, 2018.

[23] J.J. Prevost, K. Nagothu, B. Kelley, and M. Jamshidi, “Prediction of cloud data center networks loads using stochastic and neural models,” in 6th International Conference on System of Systems Engineering (SoSE), Albuquerque, NM, USA, 2011, pp. 276-281.

[24] R. Cao, Z. Yu, T. Marbach, J. Li, G. Wang, X. Liu, “Load prediction for data centers based on database service,” in IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, Japan, 2018, pp. 728-737.

[25] J. Kumar, and A.K. Singh, “Dynamic resource scaling in cloud using neural network and black hole algorithm,” in 5th International Conference on Eco-friendly Computing and Communication Systems (iECCCS), Bhopal, India, 2016, pp. 63-67.

[26] J. Kumar, A. K. Singh, and R. Buyya, “Self directed learning based workload forecasting model for cloud resource management,” Information Sciences, vol. 543, pp. 345-366, 2021.

[27] X. Tang, “Large-scale computing systems workload prediction using parallel improved LSTM neural network,” IEEE Access, vol. 8, pp. 53-67, 2019.

[28] A.K. Singh, D. Saxena, J. Kumar, and V. Gupta, “A quantum approach towards the adaptive prediction of cloud workload,” IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 12, pp. 2893-2905, 2021.

[29] M. Ghobaei-Arani, R. Khorsand, and M. Ramezanpour, “An autonomous resource provisioning framework for massively multiplayer online games in cloud environment,” Journal of Network and Computer Applications, vol. 142, pp. 76-97, 2019.

[30] M. Amiri, M. R. Feizizadeh, and L. Mohammad-Khanli, “IDS fitted Q improvement using fuzzy approach for resource provisioning in cloud,” Journal of Intelligent and Fuzzy Systems, vol. 32, no. 1, pp. 229-240, 2017.

[31] W. Hussain, J. M. Merigo, M. R. Raza, and H. Gao, “A new QoS prediction model using hybrid IOWA-ANFIS with fuzzy C-means, subtractive clustering and grid partitioning,” Information Sciences, vol. 584, pp. 280-300, 2022.
S. Sato, M. Sano, and Y. Sawada, “Practical methods of measuring the gene-

VOLUME 4, 2016

M.B. Kennel, R. Brown, and H.D.I. Abarbanel, “Determining embedding

J. Bi, S. Li, H. Yuan, and M. Zhou, “Integrated deep learning method for

M. Etemadi, M. Ghobaei-Arani, and A. Shahidinejad, “Resource provi-

C. Liu, C. Liu, Y. Shang, S. Chen, B. Cheng, and J. Chen, “An adaptive

A.M. Fraser, and H.L. Swinney, “Independent coordinates for strange

A. Wolf, J.B. Swift, H.L. Swinny, and J.A. Vastano, “Determining lya-

A. M. Fraser, and H. L. Swinney, “Independent coordinates for strange

D. Loye, and R. Eisler, “Chaos and transformation: Implications of

Z. Liu, “Chaotic time series analysis,” *Mathematical Problems in Engi-

F. Takens, “Detecting strange attractors in turbulence,” in *Dynamical

J. Bi, S. Li, H. Yuan, and M. Zhou, “Temporal Prediction of Multiapplication

I. Rodríguez-Fdez, A. Canosa, M. Mucientes, and A. Bugarín, “STAC:

C. Reiss, J. Wilkes, and J.L. Hellerstein, Google Cluster-Usage Traces:

A. Wolf, J.B. Swift, H.L. Swinny, and J.A. Vastano, “Determining lya-

B. Song, Y. Yu, Y. Zhou, Z. Wang, and S. Du, “Host load prediction with

P. Singh, P. Gupta, and K. Jyoti, “TASM: Technocrat ARIMA and SVR

M. Friedman, “The use of ranks to avoid the assumption of normality

J. Kumar, and A. K. Singh, “Performance assessment of time series

W.J. Conover, *Practical nonparametric statistics,* New York: Wiley, 1999.