Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Ensemble learning based predictive framework for virtual machine resource request prediction

Jitendra Kumar[a,b,*], Ashutosh Kumar Singh[b], Rajkumar Buyya[c]

[a] *Department of Computer Engineering and Applications, GLA University Mathura, India*
[b] *Department of Computer Applications, National Institute of Technology Kurukshetra, India*
[c] *Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, VIC 3010, Australia*

A R T I C L E I N F O

Article history:
Received 27 May 2019
Revised 19 January 2020
Accepted 2 February 2020
Available online 12 February 2020

Communicated by Dr. Leandro Minku

Keywords:
Cloud computing
Resource demand prediction
Extreme learning machine
Ensemble learning
Neural networks
Blackhole

A B S T R A C T

The cloud service providers require a large number of computing resources to provide services on-demand that consume the electricity at large and leave high carbon footprints which must be minimized. A cloud system must optimally use its resources to achieve a low operational cost without degrading the quality of services. In this context, an ensemble learning based workload forecasting method is presented that uses extreme learning machines and their corresponding forecasts are weighted by a voting engine. A metaheuristic algorithm inspired by blackhole theory is used to select the optimal weights. The accuracy of the approach is tested on CPU and memory demand requests of Google cluster trace. The method is also compared with recent existing work in the literature on CPU utilization of Google cluster and PlanetLab traces. The results validate the superiority of the approach over existing methods with an improvement up to 99.20% in root mean squared error.

1. Introduction

There has been massive growth in digital content creation driven by huge upsurge in online shopping, social networking, learning, communications etc. The organizations prefer to store and process the collected data over the cloud due to its cost-effective solutions. Unlike traditional or local infrastructure, a cloud user can access the computing resources as services hosted on the Internet. The cloud offers a number of features including on-demand resources, elasticity, flexibility, mobility, and disaster recovery. Among these, elasticity is one of the most important characteristics of the cloud paradigm that allows an application to scale its resource demands anytime in its lifespan [1,2]. The resource requests for a cloud application include the number of virtual machines that are required to execute the application's task and amount of resources such as CPU cores, memory, bandwidth assigned to each virtual machine. However, the frequent changes in an application's resource demands may increase the number of movements across the servers which may cause issues such as resource under/over provisioning as well as high power consumption [3]. IBM observed in its study that mean utilization of CPU and memory corresponds to 17.76% and 77.93%, respectively [4]. Another analysis shows that the usage of CPU and memory in a Google cluster trace could not exceed 60% and 50%, respectively [5]. The under utilization of resources results in excess use of electricity which should be minimized because an active idle machine consumes over half of the peak power consumption [6]. In 2015, data centers consumed 35TWh (Tera Watt hour) electricity as per the EIA's (Energy Information Administration) report and expected to consume 95TWh by 2040 [7]. The resource utilization can be improved by minimizing the number of active physical machines. Though, finding an optimal mapping of virtual and physical machines in an ever changing resource requirements is a complex task which belongs to NP-Complete class of problems [8]. Therefore, an intelligent resource management scheme is required to improve the quality of services (QoS) as well as financial gains of service providers [9,10].

Advanced information of future demands helps in mapping the applications to the machines such that the resource utilization is improved. However, the prior estimation of workloads (resource demands and workloads are used interchangeably) is a complex and challenging task in the presence of high variability. Prediction methods are broadly classified into two categories viz. homeostatic and history based methods [11]. The homeostatic approaches predict the workload based on current information and mean of

* Corresponding author at: Department of Computer Engineering & Applications, GLA University Mathura, India.
E-mail addresses: jitendrakumar@ieee.org (J. Kumar), ashutosh@nitkkr.ac.in (A.K. Singh), rbuyya@unimelb.edu.au (R. Buyya).

https://doi.org/10.1016/j.neucom.2020.02.014
0925-2312/© 2020 Elsevier B.V. All rights reserved.

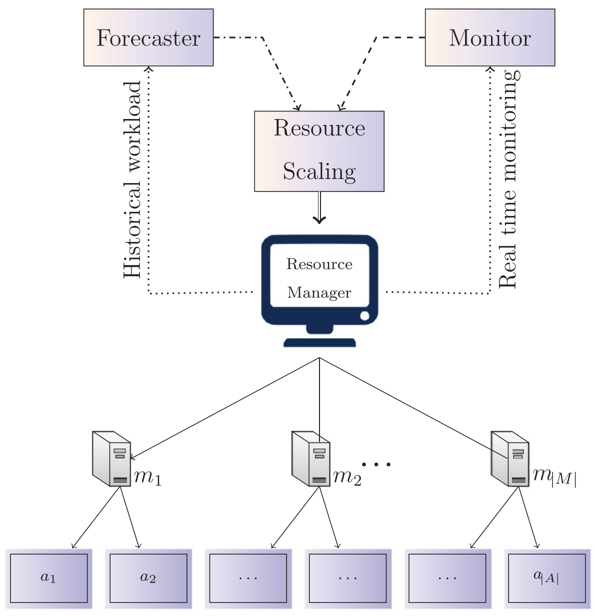


Fig. 1. Prototype of cloud system with predictive framework.

historical workload [12], while the history based methods analyze the history and predict the future workload based on the changes in recent data. The proposed work also analyzes the historical workloads to anticipate the future workload. The model considers the opinion of multiple experts to compute future workload values. The opinion of each expert is weighted by a voting engine according to their performance.

The rest of the paper is organized as follows. Section 2 defines the prediction problem. Section 3 briefly reviews the recent key contributions in machine learning based workload prediction approaches. Section 4 discusses the predictive framework methodology in detail. Section 5 explains the evaluation criteria used to measure and compare the performance of the proposed work, followed by simulation results and analytical remarks in Section 6. Finally the paper is concluded in Section 7.

2. Problem definition

A typical cloud system with workload predictor module is shown in Fig. 1. The system receives inputs from the data center's real-time state monitor and a forecaster. Based on these inputs, the resource scaling module is responsible to make efficient scaling decisions to improve the financial gains of service providers. The forecasting module is a critical component of the system, which forecasts the future trends of resource demands.

Let us consider a data center having |M|physical machines (see Eq. (1)) and each machine is equipped with|R|different resources such as CPU, memory, bandwidth, and others as shown in Eq. (2). The data center is hosting|A|applications, as in Eq. (3) and each application requests for certain amount of different resources as given in Eq. (4). Assuming, an application (a_j) is assigned to a machine (m_i) only if it can satisfy the resource demands of a_j i.e. $R^{req(a_j)} \leq R^{avl(m_i)}$. Where, $R^{avl(m_i)}$ denotes the available amount of resources at i^{th} physical machine as represented in Eq. (5). Table 1 lists out the variable notations and their corresponding description.

$$M = \{m_1, m_2, \dots, m_{|M|}\} \tag{1}$$

$$R^{m_i} = \{r_1^{m_i}, r_2^{m_i}, \dots, r_{|R|}^{m_i}\} \tag{2}$$

$$A = \{a_1, a_2, \dots, a_{|A|}\} \tag{3}$$

$$R^{req(a_j)} = \{r_1^{req(a_j)}, r_2^{req(a_j)}, \dots, r_{|R|}^{req(a_j)}\} \tag{4}$$

Table 1
List of variables.

Notation	Description
m_i	ith physical machine
$\vert M\vert$	Total physical machines in datacenter
$\vert R\vert$	Types of resources
a_j	jth application
$\vert A\vert$	Total applications hosted in data center
$R_{req(a_j)}$	Resource requested by a_j
$R_{avl(m_i)}$	Resource available on m_i
$\mathcal{Y}$	Set of workload patterns
$\boldsymbol{y_j}$	jth workload pattern input
τ_j	jth workload pattern output
$h_i^{\varepsilon_k}$	ith hidden neuron of ε_k
$\hat{\tau}_j^{\varepsilon_k}$	Forecast for jth workload pattern by ε_k
$\mathcal{E}$	Set of k ELMs (ε_k)
n	Number of input neurons
p	Number of hidden neurons
q	Number of ouput neurons
$\mathcal{S}$	Population
s^*	Blackhole solution
ρ	Radius of event horizon area of blackhole
δ	distance of a solution from blackhole
L	Max lag terms
sl	Significant lag terms
T_ρ	Significant autocorrelation threshold
Ξ_{tr}	Training error
$T_{\Xi_{tr}}$	Training error threshold
Ξ_{ts}	Test error
$T_{\Xi_{ts}}$	Test error threshold

$$R^{avl(m_i)} = \{r_1^{avl(m_i)}, r_2^{avl(m_i)}, \dots, r_{|R|}^{avl(m_i)}\} \tag{5}$$

$$\hat{R}_{n+1}^{req(a_j)} = f(R_n^{req(a_j)}, R_{n-1}^{req(a_j)}, \dots, R_1^{req(a_j)}) \tag{6}$$

The resource request sequence of a_j represented as $[R_1^{req(a_j)}, R_2^{req(a_j)}, \dots, R_t^{req(a_j)}]$ is monitored and stored over time, where $R_t^{req(a_j)}$ represents the amount of resources requested by a_j at time instance t. The forecaster examines n instances of its past behavior which form a sequence called learning window to estimate the next request instance as shown in Eq. (6). The task of the framework is to forecast the expected amount of resources to be requested by a_j at a regular interval and duration between each forecast is referred as prediction window. The objective of the proposed predictive method is to minimize the difference between $\hat{R}_{n+1}^{req(a_j)}$ and $R_{n+1}^{req(a_j)}$ for each time instance.

3. Recent key contribution

In this section, the forecasting models developed using machine learning algorithms are discussed only because the underlying methodology of the proposed approach belongs to the same domain. However, an extensive review of resource management approaches can be found in [13–17].

3.1. Neural network based approaches:

An evolutionary neural network was used for workload prediction [18] that implements particle swarm optimization, differential evolution, and covariance matrix adaptation evolutionary strategy learning algorithms and compares their performance. The neural network based prediction approach for estimation of workloads is proposed in [19] that uses an adaptive approach of differential evolution for network training. The adaptive nature of the algorithm reduces the overhead of parameter tuning. A Bayesian approach was developed to predict the VM workloads in [20] that uses demand forecasts to determine whether an application is CPU and/or memory intensive and resources are configured accordingly. The dynamic resource provisioning had been achieved using predictive approaches based on constraint programming and neural network [21]. A predictive resource allocation scheme was designed to improve the performance of cloud system [22] which works well with its error tolerance capability if predicted information is inaccurate. The proactive workload management is achieved by an intelligent workload factoring [23] that separates the workloads in two different categories viz. base crowd and flash crowd based on different components of applications and detects the items to factor the incoming requests in the context of data and volume. A predictive framework for resource provisioning to optimize the energy consumption was developed in [24] which predicts the VMs along with their resource demands expected to arrive in future. It also provide the information of required physical resources to fulfill the future demands with optimized use of energy. The similar work is reported in [25] that optimizes the resource utilization, energy consumption, and secure allocation. The predictive approach involves the use of clustering and Wiener filter. The fuzzy theory had been applied in workload prediction of cloud servers [26]. The approach keeps track of historical and current CPU utilization to forecast future demands. It can also estimate the available resources by predicting the resource utilization of physical machines. A study on different servers' resource utilization was carried out in [27] that observed the misalignment of patterns in time. Moreover, a set of algorithms were developed to refine the utilization patterns to reduce the over provisioning for resources. Genetic algorithm based workload predictive resource management was proposed in [28] which improves the average utilization and energy consumption. The method is a combination of prediction and placement approaches. The evolutionary neural networks have been a great choice for solving complex optimization problems including predictive analytics [29]. However, they need high training time.

3.2. *Deep learning based approaches*

A framework responsible for resource allocation and power management based on deep learning is presented in [30] which incorporates a forecaster that provides the estimated workload information to the power manager. The forecaster module is developed using long short term memory (LSTM) recurrent neural network. The power manager takes the forecasts and the current state information into account to decide further actions. Qiu et al. used a set of restricted Boltzmann machines arranged in a layered approach along with a regression layer to develop a predictive method to estimate VM workloads [31]. The workload prediction mechanism based on LSTM networks had been explored in [32] which used four LSTM units to improve the quality of forecasts. An efficient workload prediction model based on deep learning was presented in [33] that converts the weight vectors into canonical polyadic decomposition to compress the model attributes. In addition, the work also proposed a learning methodology based on back propagation for the training of the auto encoder's parameters. A deep reinforcement learning based resource management scheme was proposed in [34]. The resource manager was composed of monitor, allocator, and controller devices which were responsible for resource utilization information gathering, mapping of applications to the resource pool, and resource configuration negotiation, respectively. A prediction model that uses deep learning was proposed in [35] which analyzes the past workload information to compute the correlation among VMs and predicts the future workload information accordingly [35]. The deep learning based frameworks are encountered with the requirements of a large number of labeled examples which results in the increase in training time. Also, the selection of suitable deep learning architecture is another concern.

3.3. *Mining based approaches*

A prediction method based on sequential pattern mining was presented in [36]. The correlation between resource variables was considered in pattern extraction of applications' behavior and these patterns were used for workload forecasting on the cloud server. Further, a prediction approach based on episode mining with online learning capability was proposed in [37]. The learning method was inspired by the different categories of human memory called long term and short term memory. The long term memory was responsible to store the episodes of application behavior in a long period while the second category stores the new behavior of applications which correspond to online learning. The run-length encoding based prediction scheme for processor power management had been presented in [38] which was energy efficient and efficaciously addressed the repetitious behavior of workloads. A resource management scheme utilizing forecasting and skewness was introduced in [39] that minimizes the skewness to combine the different categories of workloads, which improves resource utilization.

3.4. *Hybrid approaches*

The predictive frameworks that employ only one forecasting model are usually able to fit a specific pattern of workloads and fail in handling the real-world traces where the pattern changes rapidly over time [40]. In such cases the resources remain over and under provisioned. Therefore, a scheme that can adapt to sudden changes becomes more useful. In this context, two online learning ensemble learning approaches for workload prediction are developed [41]. A workload prediction scheme based on using weighted random forest was developed in [42] which also introduced an error correction mechanism. The predictive approach employs a set of the random forest where each of them was trained on the different training set. The forecasts of each model were weighted to compute the final forecast. A workload prediction framework 'CloudInsight' was developed using a set of predictors [43]. It combines 8 different prediction methods from machine learning, time series, and regression classes to improve the accuracy of forecasts. The support vector machines were used to predict the workload sequences in [44]. The authors used particle swarm optimization to optimize the model parameters. A number of approaches have been proposed and utilized to forecast the workloads on the servers.

It was observed that the above mentioned works were unable to model and forecast the different type of data traces as they were developed and trained for a specific type of workloads. Therefore, the combination of various methods was used to model and forecast the workloads. But the existing hybrid forecasting methods such as [45–47] are generally a combination of several machine learning models which suffered from a high computational complexity.

The above discussion concludes that most of the predictive frameworks use a single approach or model to anticipate future workload and their accuracy tends to drop down as the pattern of workloads changes. The hybrid approaches have been proposed to address this issue but unfortunately, they suffer from high computational complexity in training. In this paper, an ensemble approach is presented for cloud datacenter workload estimation to addresses the aforementioned workload issues. The framework creates an ensemble using ELMs that are fast and computationally efficient learners. The predictions of individual networks are weighted using a voting engine that optimizes the weights using blackhole algorithm. The blackhole algorithm is used due to the fact that it overcomes the issue of parameter tuning in evolutionary algorithms as it does not use any additional parameter except the common variables such as population size and

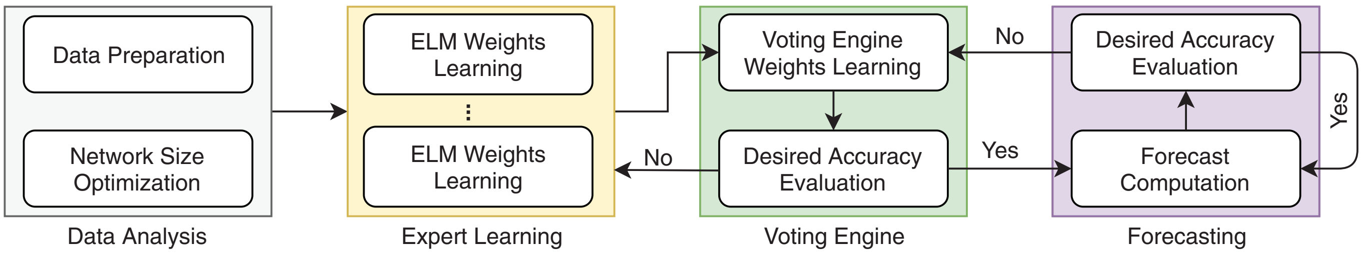


Fig. 2. Block diagram of proposed predictive framework.

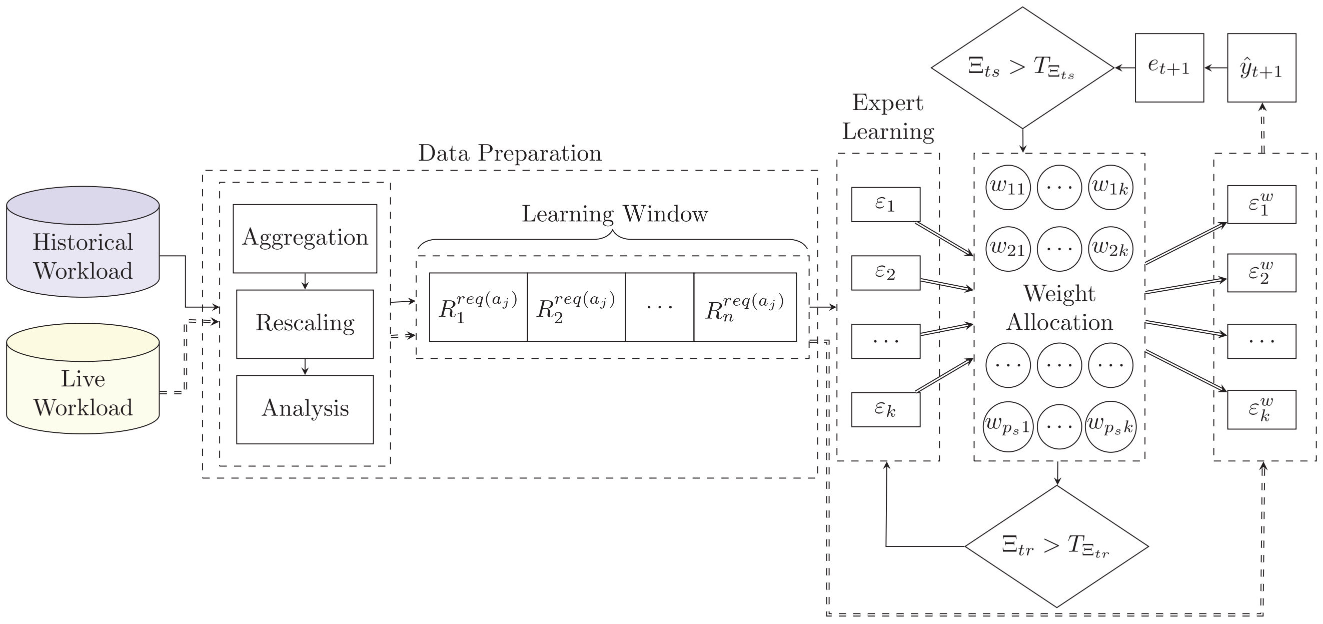

Historical Workload
Live Workload
Data Preparation
Aggregation
Rescaling
Analysis
Learning Window
$R_1^{req(a_j)}$ $R_2^{req(a_j)}$ $\cdots$ $R_n^{req(a_j)}$
Expert Learning
ε_1 ε_2 $\cdots$ ε_k
w_{11} $\cdots$ w_{1k}
w_{21} $\cdots$ w_{2k}
Weight Allocation
w_{p_s1} $\cdots$ w_{p_sk}
ε_1^w ε_2^w $\cdots$ ε_k^w
$\Xi_{ts} > T_{\Xi_{ts}}$
e_{t+1}
$\hat{y}_{t+1}$
$\Xi_{tr} > T_{\Xi_{tr}}$

Fig. 3. Detailed workflow of proposed predictive framework.

dimension [48,49]. The efficacy of the proposed approach is tested on the benchmark datasets however, it can be applied on any data trace due to the fact that the framework is enabled with the ability of selecting the suitable network structure by analysing the characterstics of data trace under consideration.

4. Prediction framework

This section presents the proposed predictive model that employs an ensemble approach to improve the quality of forecasts. The framework's block diagram is given in Fig. 2. There are three key modules namely data analysis, expert learning, and voting engine which are responsible for preprocessing, base predictors weights learning, and optimizing the weights corresponding to each base prediction, where arrows depict the flow of information from one step/module to another step/module. The legends on the arrows (yes/no) controls the conditional execution of next step on the basis of accuracy obtained.

The complete workflow of the model is shown in Fig. 3. The data preparation involves the aggregation of resource demands per time unit, rescaling the aggregated values in [0, 1] range using min–max normalization followed by an estimation of a number of previous workload instances that affect the future values. The prepared data is fed into each expert to get an estimation of the upcoming workload information. Further, each expert's prediction is weighted to compute the final outcome of the predictive module and these weights were optimized using a metaheuristic based weight allocation process. If the forecast error (Ξ_{tr}) during model training is beyond the threshold ($T_{\Xi_{tr}}$), the experts learn network weights and the rest of the procedure is executed again. Similarly, if system predicts the workload on live data with error (Ξ_{ts}) greater than the tolerance level ($T_{\Xi_{ts}}$), only weight allocation scheme refines the weights for base experts.

4.1. Ensemble expert learning using ELM

An ensemble approach involves the use of multiple prediction models to forecast the estimated future outcome of an event and each of them is commonly referred to as an expert or base predictor. The final outcome of an ensemble is computed by combining the forecasts of each expert using a voting engine. The conceptual architecture of an ensemble based predictive approach is depicted in Fig. 4.

The proposed framework employed k multilayer neural networks ($\mathcal{E} = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_k\}$) as base experts each composed with n, p, and q number of neurons in input, hidden, and output layers correspondingly along with $\zeta(\bullet)$ activation function at hidden layer. The experts are trained using extreme learning machine algorithm that select the hidden layer weights randomly and compute the output layer weights analytically [50]. An ELM solves a general linear system to find out the weights of synaptic connections between hidden and output neurons. ELMs are well known for their speed and they can universally approximate any continuous function [51].

Given a set of workload patterns $\mathcal{Y} = \{(\boldsymbol{y_j}, \tau_j) | \boldsymbol{y_j} \in \mathbb{R}^n, \tau_j \in \mathbb{R}\}$, where $\boldsymbol{y_j} = [R_j^{req(a_j)}, R_{j+1}^{req(a_j)}, \ldots, R_{j+n-1}^{req(a_j)}]^T$ and $\tau_j = R_{j+n}^{req(a_j)}$ constituted the j^{th} workload pattern that contained the previous

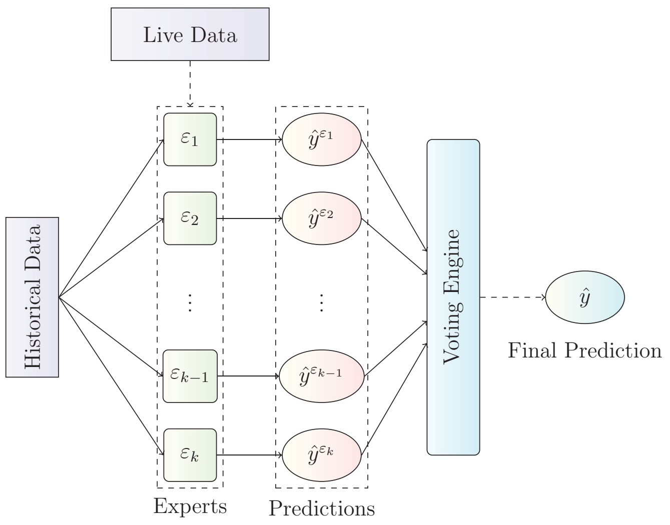


Fig. 4. A conceptual view of ensemble stacking approach.

resource request instances of the length of learning window and actual amount of resource requested at next time instant (see Eq. (7)). Let the weights of synaptic connections between ith hidden neuron ($h_i^{\varepsilon_k}$) and input neurons of ε_k are represented as $\boldsymbol{\omega}_i^{\varepsilon_k} = [\omega_{1i}^{\varepsilon_k}, \omega_{2i}^{\varepsilon_k}, \dots, \omega_{ni}^{\varepsilon_k}]$ and the bias connection to $h_i^{\varepsilon_k}$ is weighted with $\beta_i^{\varepsilon_k}$. The output of $h_i^{\varepsilon_k}$ on $\boldsymbol{y}_j$ can be modeled as $\hbar_i^{\varepsilon_k}(\boldsymbol{y}_j) = \zeta(\boldsymbol{\omega}_i^{\varepsilon_k} \cdot \boldsymbol{y}_j + \beta_i^{\varepsilon_k})$, which is further provided to output neurons. If the weights of connections between $h_i^{\varepsilon_k}$ and output neurons were represented as $\tilde{\boldsymbol{\omega}}_i^{\varepsilon_k} = [\tilde{\omega}_{i1}^{\varepsilon_k}, \tilde{\omega}_{i2}^{\varepsilon_k}, \dots, \tilde{\omega}_{iq}^{\varepsilon_k}]^T$, the j^{th} pattern forecast using ε_k can be mathematically modeled as given in Eq. (8).

$$\mathcal{Y} = \begin{bmatrix} R_1^{req(a_j)} & R_2^{req(a_j)} & \cdots & R_n^{req(a_j)} \\ R_2^{req(a_j)} & R_3^{req(a_j)} & \cdots & R_{n+1}^{req(a_j)} \\ \vdots & \vdots & \ddots & \vdots \\ R_m^{req(a_j)} & R_{m+1}^{req(a_j)} & \cdots & R_{n+m-1}^{req(a_j)} \end{bmatrix} \begin{bmatrix} R_{n+1}^{req(a_j)} \\ R_{n+2}^{req(a_j)} \\ \vdots \\ R_{n+m}^{req(a_j)} \end{bmatrix} \tag{7}$$

$$\hat{\tau}_j^{\varepsilon_k} = \sum_{i=1}^{p} \tilde{\boldsymbol{\omega}}_i^{\varepsilon_k} \times \zeta(\boldsymbol{\omega}_i^{\varepsilon_k} \cdot \boldsymbol{y}_j + \beta_i^{\varepsilon_k}); \qquad \forall j \in \{1, 2, \dots, m\} \tag{8}$$

The forecasts of an expert on $j = (1, 2, \dots, m)$ patterns (obtained from Eq. (8)) can be written as $\hat{\boldsymbol{\tau}}^{\varepsilon_k} = \sum_{i=1}^{p} \tilde{\boldsymbol{\omega}}_i^{\varepsilon_k} \times \hbar_i^{\varepsilon_k}(\boldsymbol{y}_j) = \tilde{\boldsymbol{\omega}}^{\varepsilon_k} \times \boldsymbol{\hbar}^{\varepsilon_k}$, where $\boldsymbol{\hbar}^{\varepsilon_k}$ is the output of hidden layer as shown in Eq. (9). The forecasts can be approximated with no error, if the system can find $\tilde{\boldsymbol{\omega}}^{\varepsilon_k}$, $\boldsymbol{\omega}^{\varepsilon_k}$, and β^{ε_k} which produce forecasts such that $\hat{\tau}_j^{\varepsilon_k} = \tau_j$ holds true for all m patterns. The minimum forecast error can be mathematically formulated as $\min_{\tilde{\boldsymbol{\omega}}^{\varepsilon_k}} ||\tilde{\boldsymbol{\omega}}^{\varepsilon_k} \boldsymbol{\hbar}^{\varepsilon_k} - \boldsymbol{\tau}||^2$.

$$\boldsymbol{\hbar}^{\varepsilon_k} = \begin{bmatrix} \boldsymbol{\hbar}^{\varepsilon_k}(y_1) \\ \boldsymbol{\hbar}^{\varepsilon_k}(y_2) \\ \vdots \\ \boldsymbol{\hbar}^{\varepsilon_k}(y_m) \end{bmatrix} = \begin{bmatrix} \hbar_1^{\varepsilon_k}(y_1) & \hbar_2^{\varepsilon_k}(y_1) & \cdots & \hbar_p^{\varepsilon_k}(y_1) \\ \hbar_1^{\varepsilon_k}(y_2) & \hbar_2^{\varepsilon_k}(y_2) & \cdots & \hbar_p^{\varepsilon_k}(y_2) \\ \vdots & \vdots & \ddots & \vdots \\ \hbar_1^{\varepsilon_k}(y_m) & \hbar_2^{\varepsilon_k}(y_m) & \cdots & \hbar_p^{\varepsilon_k}(y_m) \end{bmatrix} \tag{9}$$

Unlike gradient or population based learning approaches, an ELM learns the $\tilde{\boldsymbol{\omega}}^{\varepsilon_k}$ i.e. synaptic connection weights between hidden and output neurons by solving a general linear system. In order to obtain the approximated values of $\tilde{\boldsymbol{\omega}}^{\varepsilon_k}$, an ELM assigns the input and bias weights randomly (in [0, 1] range for our experiments) and finds least-square solution for general linear system $\boldsymbol{\hbar}^{\varepsilon_k} \cdot \tilde{\boldsymbol{\omega}}^{\varepsilon_k} = \boldsymbol{\tau}$ as shown in Eq. (10), where $\tilde{\boldsymbol{\Omega}}^{\varepsilon_k}$ and $\boldsymbol{\hbar}^{\varepsilon_k^\dagger}$ are the least square solutions with minimum norm and uniqueness, and Moore–Penrose generalized inverse of a matrix $\boldsymbol{\hbar}^{\varepsilon_k}$, respectively.

$$\tilde{\boldsymbol{\Omega}}^{\varepsilon_k} = \boldsymbol{\hbar}^{\varepsilon_k^\dagger} \boldsymbol{\tau} \tag{10}$$

After approximating the synaptic connection weights of each expert, the forecasts on training data are weighted using a voting engine. The weights (α^{ε_k}) are assigned using a heuristic based weight allocation scheme. The expert learning module aims to train k base predictors on training data as shown in lines 3–5 of Algorithm 2.

4.2. Expert architecture selection

The proposed framework creates an ensemble of base predictors that are designed exploiting the neural network. The aim of this step is to find the best suitable structure of the network to attain better performance. The architecture (number of layers and number of nodes in each layer) of a network is critical to its performance. The predictive approach used k networks with n, p, and q neurons in each layer as shown in Fig. 5. Since we kept a fixed number of network layers and neurons in the output layer, we optimized the number of neurons in both the input and hidden network layer only.

The model selects the number of input neurons (n) based on the data characteristics as shown in Algorithm 1. Since resource demand traces are indexed in time, it can be considered as time series objects and the model determines the number of lags with significant autocorrelation to optimally select the input neurons.

Algorithm 1 Input node selection.

```
Input: $R^{req(a_j)}$, $L$, $T_\varrho$, $d_o$
Output: $n$
1: Initialize: $d = 1$
2: Compute autocorrelation ($\varrho$) of data trace up to $L$ lags using Eq. (11)
3: Determine the lags with significant autocorrelation using Eq. (12)
4: if ($sl \neq 0$ && $sl \neq L$) then
5:     $n = sl$
6: else
7:     if ($d \leq d_o$) then
8:         Differentiate data trace
9:         $d = d + 1$
10:        Repeat STEP 2 on differenced trace
11:    else
12:        $n = L$
13:    end if
14: end if
15: return $n$
```

The algorithm requires $R^{req(a_j)}$, L, T_ϱ, d_o as input, where each term represents the resource request trace, maximum time lags to compute autocorrelation, threshold to decide the significant autocorrelation, and maximum number of terms used for differentiation of data trace, respectively. Let $\varrho_{R^{req(a_j)}}$ is the autocorrelation of resource requests trace of a_j computed by line 2, where `autocorr` is a MATLAB function that computes the autocorrelation of a univariate series. The autocorrelation measures and explains the internal association among time series observations. The value of L was chosen based on experimental analysis and can be extended to any possible number. The approach analyzes the autocorrelation in a data trace to select the optimal number of input neurons of base experts. The approach finds the presence of autocorrelation in the original data trace and differentiated data trace up to 40 time Lags. It was observed that the significant (higher than threshold) autocorrelation was present up to 40 time

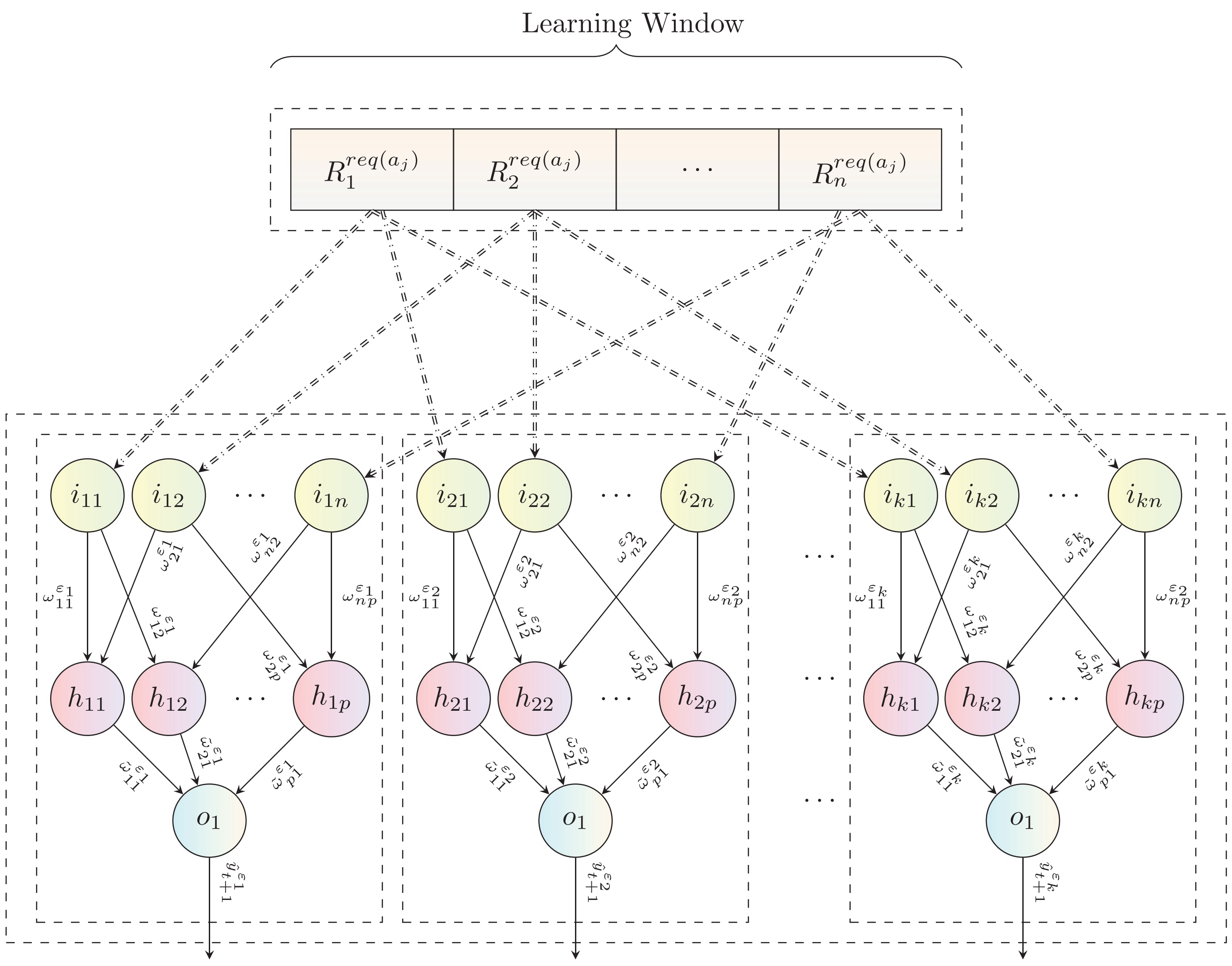

Learning Window
$R_1^{req(a_j)}$
$R_2^{req(a_j)}$
$R_n^{req(a_j)}$
i_{11} i_{12} i_{1n}
h_{11} h_{12} h_{1p}
o_1
i_{21} i_{22} i_{2n}
h_{21} h_{22} h_{2p}
i_{k1} i_{k2} i_{kn}
h_{k1} h_{k2} h_{kp}
$\hat{y}_{t+1}^{\varepsilon_1}$
$\hat{y}_{t+1}^{\varepsilon_2}$
$\hat{y}_{t+1}^{\varepsilon_k}$

Fig. 5. An ensemble of extreme learning machines.

Lags in original traces but in differentiated data traces the significant autocorrelation was present up to a very small time lags. For instance, as shown in Fig. 6, significant autocorrelation was present up to 3 time lags in differentiated traces. Therefore, for experimental purpose we put a limit on L up to 40 to ensure the analysis of data in depth. However, it may be adjusted as per the data trace characteristics. As shown in the algorithm that $\varrho_{R^{req(a_j)}}$ was analyzed by applying Υ operation to find the last lag with significant autocorrelation (line 3). The last lag with significant autocorrelation (*sl*) is defined to be the number of input neurons. If every lag has correlation higher than the threshold, the data is differentiated and the procedure is repeated upto d_o (difference order) times to obtain the number of input neurons (lines 4–14).

$$\varrho_{R^{req(a_j)}} = \texttt{autocorr}(R^{req(a_j)}, L) \tag{11}$$

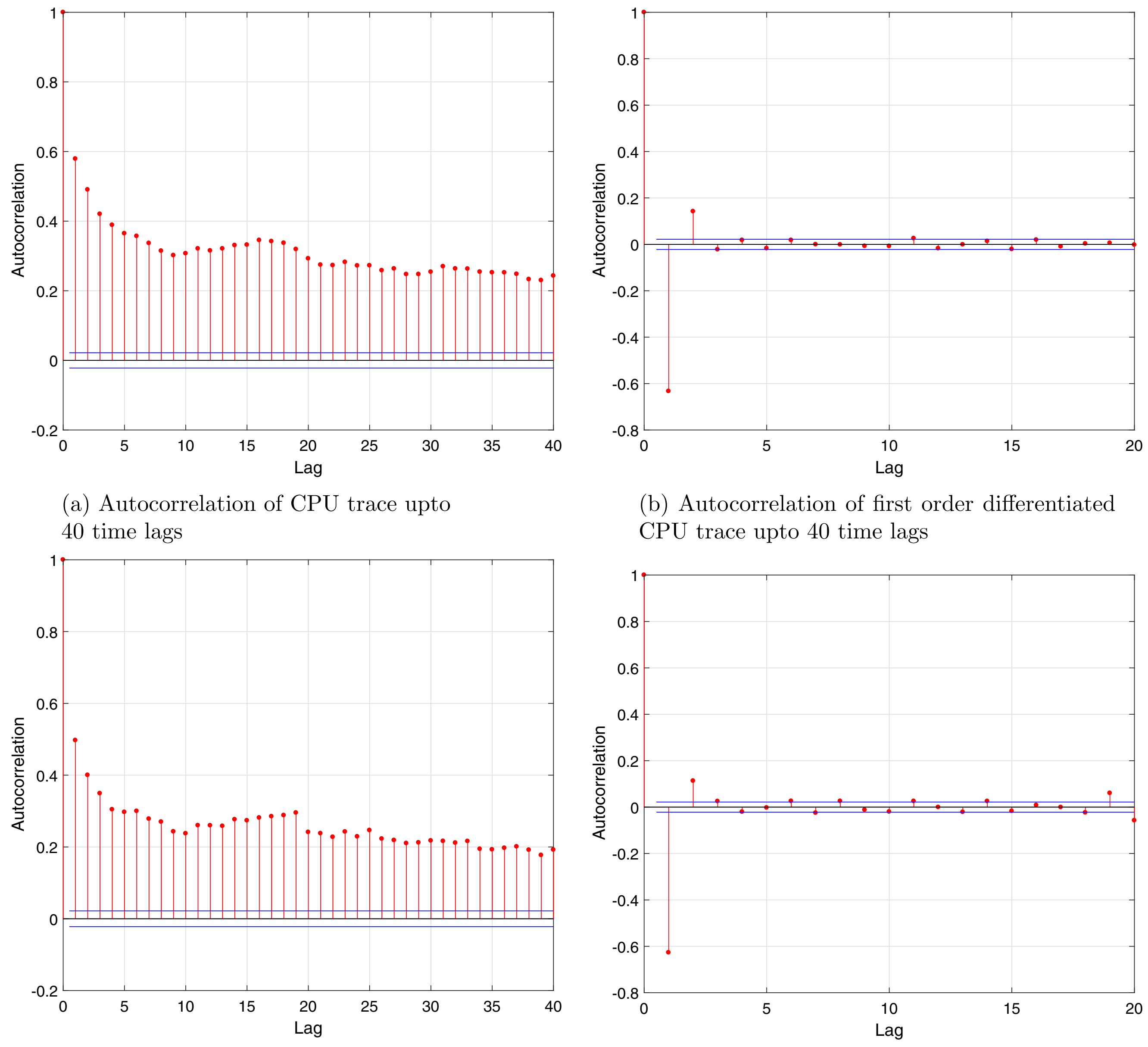
$$sl = \Upsilon((|\varrho_{R^{req(a_j)}}| \le T_\varrho), 1) \tag{12}$$

The hidden neurons are selected based on three different schemes as given in Eq. (13), where $p_{\min}$ and $p_{\max}$ represent the minimum and maximum hidden neurons, respectively. In fix scheme, the number of hidden neurons for each network are kept same as the average of $p_{\min}$ and $p_{\max}$. In linear approach, the number of hidden neurons for k^{th} network are selected as $p^{\varepsilon_k} = p_{\min} + k$ while the third approach (random) computes the hidden neurons by selecting a random number between $p_{\min}$ and $p_{\max}$. The experimental analysis reports that the linear scheme performed better (the detailed discussion is given in Section 6) and results are reported based on the selected scheme.

$$p^{\varepsilon_k} = \begin{cases} p_{min} + k, & \text{Linear} \\ randi[p_{min}, p_{max}], & \text{Random} \\ \lfloor (p_{min} + p_{max})/2 \rfloor, & \text{Fix} \end{cases} \tag{13}$$

4.3. Weight allocation scheme

The base experts of the predictive framework produce their corresponding forecasts and each of them is weighted to compute the final predicted workload. The weight allocation module aims to find the best possible weights associated with each base predictor to minimize the forecast error. A population based metaheuristic algorithm is used to optimize the weights associated with different base experts. Unlike gradient based methods where a single solution explores the search space by learning from its past experience, population based approaches allow multiple candidate solutions to explore a search space simultaneously. We selected an optimization algorithm inspired by blackhole phenomenon of the nature [52]. Unlike other learning approaches such as genetic algorithm, differential evolution and others, the blackhole optimization algorithm does not involve any parameters in learning process. The blackhole optimization algorithm (BhOA) also is a population



(a) Autocorrelation of CPU trace upto 40 time lags

(b) Autocorrelation of first order differentiated CPU trace upto 40 time lags

(c) Autocorrelation of Memory trace upto 40 time lags

(d) Autocorrelation of first order differentiated Memory trace upto 40 time lags

Fig. 6. Autocorrelation of data traces (5 min).

based method which iterates a number of times to approximate an optimal solution from search space. Let $\mathcal{S} = [s_1, s_2, \dots, s_{ps}]^T$ be a set of ps candidate solutions each encoded with k real numbers as shown in Eq. (14), where $\alpha_i^{\varepsilon_k}$ represents the weight factor of ε_k. The encoding scheme is bounded by two constraints i.e. the value must be in [0, 1] range and the sum of all values must be 1. The initialization of population is carried out using $pop\_init(ps, k)$ function that generates a population of ps stars each of size k (line 6 of Algorithm 2). It also ensures that each s_i satisfies the constraints mentioned in encoding of solutions (Eq. (14)).

$$s_i = [\alpha_i^{\varepsilon_1}, \alpha_i^{\varepsilon_2}, \dots, \alpha_i^{\varepsilon_k}] \tag{14}$$

subject to

$$\alpha_i^{\varepsilon_j} \in [0, 1] \qquad \forall j = \{1, 2, \dots, k\}$$

$$\alpha_i^{\varepsilon_1} + \alpha_i^{\varepsilon_2} + \dots + \alpha_i^{\varepsilon_k} = 1$$

A candidate solution contains k weights each associated with one of the base expert's forecasts. The final forecast is obtained by computing the weighted sum of base expert's individual forecasts as given in Eq. (15). Therefore, the fitness of candidate solutions is measured using root mean squared error on m labelled data samples as mentioned in Eq. (16), where $\hat{y}_t^{s_i}, y_t$ are the forecast produced by s_i and actual workload at time t, respectively (line 7, Algorithm 2).

$$\hat{y}_t^{s_i} = \sum_{j=1}^{k} \alpha_i^{\varepsilon_j} \times \hat{y}_t^{\varepsilon_j} \tag{15}$$

$$f(s_i) = \frac{1}{m} \sum_{t=1}^{m} (\hat{y}_t^{s_i} - y_t)^2 \tag{16}$$

Since the system aims to minimize the forecast error, the candidate solution with least error is considered as blackhole (s^*) and

Algorithm 2 Operational summary of predictive framework.

```
1: Select input nodes using Algorithm 1
2: [$\mathcal{Y}$] = Prepare input data according to $n$
3: for each expert ($\varepsilon_k$) do
4:     $\omega^{\varepsilon_k} = rand(n, p+1)$;   $\tilde{\boldsymbol{\Omega}}^{\varepsilon_k} = \boldsymbol{\hbar}^{\varepsilon_k \dagger} \boldsymbol{\tau}$
5: end for
6: $S = pop\_init(ps, k)$
7: $f(s_i) = \frac{1}{m}\sum_{t=1}^{m}(\hat{y}_t^{s_i} - y_t)^2$
8: $s^* = \min_{i=1,2,\dots,ps} f(s_i)$
9: for each iteration do
10:    for each star ($s_i$) do
11:        $s_i' = s_i + rand \times (s^* - s_i)$;   $f(s_i') = \frac{1}{m}\sum_{t=1}^{m}(\hat{y}_t^{s_i} - y_t)^2$
12:    end for
13:    Update Blackhole $s^*$
14:    Compute radius of blackhole horizon $\rho = \frac{f(s^*)}{\sum_{i=1}^{ps} f(s_i)}$
15:    for each star ($s_i$) do
16:        $\delta_{s_i} = f(s^*) - f(s_i)$
17:        $s_i = \begin{cases} \Psi & \text{if } \delta_{s_i} <= \rho \text{ and } s_i \neq s^*, \\ s_i & \text{otherwise} \end{cases}$
18:    end for
19: end for
20: Compute $\Xi_{tr}$
21: if $\Xi_{tr} > T_{\Xi_{tr}}$ then
22:    Goto Step 3
23: end if
24: Forecast the future workload $\hat{R}_{t+1}^{req(a_j)}$ and Compute $\Xi_{ts}$
25: if $\Xi_{ts} > T_{\Xi_{ts}}$ then
26:    Goto Step 6
27: end if
```

can be obtained as $s^* = \min_{i=1,2,\dots,ps} f(s_i)$ (line 8, Algorithm 2). Further, the s^* guides the other solutions (s_i) to explore the search space in better direction by updating their position $s_i' = s_i + rand \times (s^* - s_i)$ where $i \in \{1, 2, \dots, ps\}$ and $s_i \neq s^*$. The candidate solutions are updated iteratively and their updated positions are evaluated during the course of optimization as shown in lines 10–12 of operational summary. Next, the blackhole is updated if a better solution is discovered (line 13, Algorithm 2). The blackhole solution mimics the behavior of blackholes in the nature i.e. anything that comes into a surrounded region of the blackhole gets collapsed. This region is termed as event horizon area and it is defined by its radius (ρ) that is computing by line 14 of Algorithm 2.

In order to detect whether s_i is reached into the bounded area or not, we compute its distance from the s^* as $\delta_{s_i} = f(s^*) - f(s_i)$. If any solution ($s_i$) enters into the horizon region, the Ψ operation is applied over it (see Eq. (17)) where the candidate solution is observed and new candidate is popped up to keep the population size uniform throughout the simulation (lines 16 and 17 of Algorithm 2). We repeat this iterative process until the approximated weights are not found that are used to compute the final forecast of the predictive framework. These workload anticipations are provided to the resource scaling module as shown in Fig. 1 that acts on these values along with current state of data center to scale in or out resources for improved utilization, power consumption, quality of experience and other parameters.

$$s_i = \begin{cases} \Psi & \text{if } \delta_{s_i} <= \rho \text{ and } s_i \neq s^*, \\ s_i & \text{otherwise} \end{cases} \tag{17}$$

5. Evaluation metrics

A predictive framework utilizes the forecasts in designing the movement pattern. Therefore, the quality of scaling decisions depends on the accuracy of forecasting module. A number of error metrics have been used to measure the quality of forecasts and we use mean squared error (MSE) and relative mean absolute error (RelMAE).

5.1. Mean squared error

This metric measures the forecast error by putting high penalty on large error terms. The model is considered to be more accurate if its score is closer to 0. The mathematical representation of the metric is mentioned in Eq. (18) where m is the number of data points in workload trace.

$$MSE = \frac{1}{m}\sum_{t=1}^{m}(y_t - \hat{y}_t)^2 \tag{18}$$

5.2. RelMAE

Root mean squared error metric can not be used to compare the performance of a predictive model on different data sets because it depends on the scale of data values. Therefore, a scale free measure is required to compare the performance and we used RelMAE that can be computed by Eq. (19), which is the mean absolute error normalized by the mean absolute error of some state-of-art method. Here, we considered the Naïve method which forecasts the future values equal to the last observed value.

$$RelMAE = \frac{MAE_A}{MAE_{BM}} \tag{19}$$

6. Simulation results

The experiments are executed on a server machine equipped with two Intel® Xeon® E5-2630 v4 processors with 2.20GHz clock speed. The machine is operated on 64-bit Windows Server 2012 R2 Standard and it has main memory of 128GB. The predictive framework was implemented in MATLAB 2017a and evaluated on CPU and memory resource requests data from Google cluster traces [53] which was released by Google in 2011. It incorporates the data of 29 days collected from Google's cluster cell. The traces are organized into six different tables that contain the data of 10,388 machines of Google cluster. In the experiments, we have used the traces of CPU and memory demands of VM instances hosted on the cluster. We also used PlanetLab data trace that contains the mean CPU utilization of more than 1000 virtual machines sampled over 5 minutes interval. The virtual machines are located at more than 500 places across the globe. The data was collected on 10 random days in the months of March and April of year 2011. The experiment environment involves the settings shown in Table 2.

Table 2
Experiment settings.

Term	Value
# ELMs (k)	[10, 100]
Input nodes (n)	[sl, L]
Hidden nodes (p)	[5, 50]
Pop size (ps)	20
Max iteration ($G_{\max}$)	100
Accuracy thresholds ($T_{\Xi_{tr}}$ and $T_{\Xi_{ts}}$)	0.007
Significant correlation threshold (T_{ϱ})	0.1
Training data ratio	70%

The first set of experiments were carried out on CPU demand trace 5 and 60 minutes prediction window size (PWS), where PWS defines the time interval between two consecutive forecasts. As discussed earlier, the approach computes the autocorrelation to

Table 3
Mean squared error obtained from networks of different size (CPU trace).

#ELMs	5 min			60 min		
	Linear	Random	Fix	Linear	Random	Fix
10	0.00502	**0.00477**	0.00480	0.01080	**0.00941**	0.00975
20	0.00491	**0.00472**	0.00479	0.01030	**0.00945**	0.00967
30	0.00484	**0.00473**	0.00480	0.00997	**0.00958**	0.00971
40	0.00476	**0.00472**	0.00480	0.00963	**0.00941**	0.00972
50	**0.00472**	0.00474	0.00479	**0.00936**	0.00949	0.00980
60	**0.00467**	0.00474	0.00479	**0.00904**	0.00958	0.00968
70	**0.00464**	0.00474	0.00479	**0.00884**	0.00956	0.00972
80	**0.00460**	0.00476	0.00479	**0.00860**	0.00961	0.00978
90	**0.00456**	0.00475	0.00479	**0.00832**	0.00948	0.00976
100	**0.00453**	0.00479	0.00480	**0.00817**	0.00957	0.00976

Table 4
Mean squared error obtained from networks of different size (Memory trace).

#ELMs	5 min			60 min		
	Linear	Random	Fix	Linear	Random	Fix
10	0.000713	**0.000672**	0.000692	0.008980	0.007670	**0.007490**
20	0.000705	**0.000674**	0.000690	0.008190	0.007530	**0.007470**
30	0.000696	**0.000683**	0.000693	0.007750	**0.007270**	0.007380
40	0.000686	**0.000684**	0.000692	**0.007300**	0.007330	0.007390
50	**0.000678**	0.000681	0.000693	**0.007120**	0.007340	0.007460
60	**0.000671**	0.000686	0.000692	**0.006840**	0.007360	0.007460
70	**0.000662**	0.000685	0.000693	**0.006640**	0.007050	0.007460
80	**0.000659**	0.000686	0.000692	**0.006460**	0.007230	0.007430
90	**0.000651**	0.000684	0.000692	**0.006210**	0.007300	0.007430
100	**0.000646**	0.000686	0.000693	**0.006000**	0.007340	0.007420

find out the number of input neurons. Any value of autocorrelation larger than T_Q is considered to be significant and number of input nodes are selected accordingly. Since the autocorrelation of non differentiated CPU trace aggregated on 5 min interval is higher than T_Q for all time lags as shown in Fig. 6(a), the data trace is differentiated and autocorrelation is recomputed. The first-order differentiation of CPU request trace reduced the autocorrelation significantly as depicted in Fig. 6(b). Therefore, the model selects 3 input neurons in this case as the autocorrelation after time lag 3 tends to zero. Similarly, the number of input neurons can be optimized for other data traces as well. Therefore, for the experimental purpose we put a limit on L up to 40 that may be adjusted as per the data trace characteristics.

Further, we compared the effect of hidden neuron selection strategies on the forecast accuracy of CPU trace as shown in Table 3, where best values are highlighted using bold face. We observed that the random scheme gives better accuracy up to 40 ELMs while linear method outperforms the other two methods afterward. The fix scheme contains the same architecture for all networks and random scheme selects the random number of hidden neurons in the allowed range between $p_{\min}$ and $p_{\max}$. But linear selection method increases the number of hidden neurons as k increases and each network has unique architecture. Therefore, the linear method outperforms the other two approaches after a certain number of networks. Since linear method achieves better, the results obtained using linear hidden neurons selection scheme are reported and considered for comparison with prior work. Similar to CPU data trace, we also experimented with memory demand traces of the Google cluster trace. Figs. 6(c) and (d) shows the autocorrelation of the series aggregated on 5 min interval. Similar to CPU trace, we observed the effect of a different number of hidden neurons on the forecast accuracy of the memory trace. Table 4 confirms that the networks based on linear selection method produced better forecasts. The forecast results reveal similar accuracy trend as of CPU trace over different prediction window intervals.

Table 5
Friedman test results.

Data	Statistic	p-value	Result
CPU	11.15873	0.00015	H_0 is rejected
Memory	7.11684	0.00237	H_0 is rejected

Table 6
Friedman test ranks.

Approach	Ranks	CPU memory
Random	1.60	1.75
Linear	1.70	1.65
Fix	2.70	2.60

Table 7
Finner post-hoc analysis result.

	CPU			Memory		
	Statistic	Adjusted p-value	Result (H_0)	Statistic	Adjusted p-value	Result (H_0)
Fix vs linear	3.162	0.002	Rejected	3.004	0.007	Rejected
Fix vs random	3.478	0.001	Rejected	2.687	0.010	Rejected
Linear vs random	0.316	0.751	Accepted	0.316	0.751	Accepted

Table 8
Forecast accuracy of the proposed model measured using MSE and RelMAE.

PWS (min)	Data	MSE	RelMAE
5	CPU	0.00453	0.86000
	Memory	0.00064	0.85900
30	CPU	0.00893	0.86000
	Memory	0.00707	0.86900
60	CPU	0.00817	0.86500
	Memory	0.00600	0.84500

A statistical test is conducted using the Friedman test with Finner post hoc analysis to validate the results. The Friedman test considers a null hypothesis (H_0) that assumes that there is no significant difference in the results of different approaches. Finner post hoc analysis helps in analyzing the pairwise performance of the models. The test is conducted with a significance level of 0.05 and on STAC [54] web platform. Table 5 shows that the Friedman test rejects the H_0 for both CPU and Memory traces that indicates the presence of a significant difference in the results. The ranks obtained through the Friedman test are shown in Table 6. It can be observed that the best rank was obtained by random and linear on CPU and Memory data traces, respectively. The Finner analysis shows that a significant difference is present in the first two pairs (Fix vs Linear and Fix vs Random) as shown in Table 7. However, the analysis observed no significant difference between linear and random approaches for both traces.

Table 8 shows the performance of the proposed approach on CPU trace on both metrics. The accuracy of forecasts drops downs as the prediction interval increases due to the fact that the most recent history contributes the most in forecasting the next event. The length between most recent available history and forecast point increases as prediction interval increases. Therefore, the accuracy of long term forecasts drops down with respect to the short term horizon.

The forecast accuracy of the proposed approach is compared with state-of-art approaches to show its efficacy [33,55]. Tables 9 and 10 compare the performance with auto regressive and moving average (ARIMA), support vector machines (SVR) and Deep Learning based models. Since Baldan et al. [55] used mean CPU utilization of the Google cluster trace to evaluate the performance of various forecasting approaches, the experiments on the same data trace are also conducted to compare the performance with

Table 9
Forecast accuracy comparison on CPU utilization of Google cluster trace using RelMAE.

PWS (min)	ARIMA [55]	SVR [55]	Proposed
5	0.9720	0.9690	**0.3281**
60	0.9210	–	**0.0156**

Table 10
Forecast accuracy comparison on CPU utilization of PlanetLab trace using RMSE.

PWS (min)	Deep learning [33]	Proposed
5	9.1700	**0.0731**
30	10.3000	**0.0956**
60	9.9700	**0.1030**

two promising forecasting approaches called auto regressive and moving average (ARIMA) and support vector machines (SVR). An improvement is observed in the forecast accuracy of the proposed framework. Since the objective of the framework is to minimize the prediction error, the performance is compared by computing the percentage decrease in the forecast error. This is achieved by computing the difference between the forecast error of the state-of-art and proposed methods. This difference is divided by the forecast error of the state-of-art method and multiplied by 100. The proposed approach observed the percentage error decrease by 66.24% and 98.31% for 5 and 60 minute forecasts respectively on comparing the performance with ARIMA. On comparing the performance with SVR based predictive model, we noticed the percentage error decrease up to 66.14%. Similarly, the performance of the proposed approach on mean CPU utilization of PlanetLab data trace is compared with [33] using the root mean squared error. It can be seen that the forecast root mean squared error is improved by percentage error decrease up to 99.20%. The results convey that the proposed approach produces forecasts with higher accuracy.

7. Conclusion

The cloud computing allows us to access the resources such as CPU, memory, disk and others over the Internet. Today, it has become the state-of-art hosting platform for government, social media and industrial applications. The number of applications being hosted over the cloud is increasing rapidly due to its characteristics including accessibility, elasticity and on-demand. However, the cloud paradigm suffers from low resource utilization and high power consumption which must be addressed on priority. In this, paper we have proposed a forecasting approach for intelligent cloud resource management to reduce the resource wastage, energy consumption and carbon footprints. The proposed method exploits the fast learning capacity of extreme learning machines to improve the training time. The scheme is tested on benchmark data sets and compared with existing state-of-art forecasting techniques. On comparing its performance with the recent methods, the analysis shows the superiority of the approach in terms of forecast accuracy. The forecasts with improved quality will ensure the low operational cost of the cloud system by achieving advancement in resource management decisions. Inspite of above mentioned benefits of the proposed model, the model has two major limitations. First, it manually selects the number of networks. Second, the number of hidden nodes in each network are selected based on heuristics. These limitations can be minimized by extending the framework to automatically select the number of networks and hidden nodes in each networks. Also, multiple time series forecasting can be incorporated.

Declaration of Competing Interest

No conflict of interest exists.

Acknowledgment

The authors would also like to thank the Ministry of Electronics and Information Technology (MeitY), Govt. of India for financial support to carry out this research work. Authors are also thankful to Dr. Ankit Saxena, Department of Management, GLA University to proofread the manuscript.

References

[1] S. Pandey, L. Sammut, R.N. Calheiros, A. Melatos, R. Buyya, Scalable deployment of a ligo physics application on public clouds: workflow engine and resource provisioning techniques, in: X. Li, J. Qiu (Eds.), Proceedings of the Cloud Computing for Data-Intensive Applications, first ed., Springer New York, New York, NY, 2014, pp. 3–25.
[2] M.D. de Assunção, A.D.S. Veith, R. Buyya, Distributed data stream processing and edge computing: A survey on resource elasticity and future directions, J. Netw. Comput. Appl. 103 (2018) 1–17.
[3] W. Voorsluys, J. Broberg, S. Venugopal, R. Buyya, Cost of virtual machine live migration in clouds: A performance evaluation, in: M.G. Jaatun, G. Zhao, C. Rong (Eds.), Cloud Computing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 254–265.
[4] R. Birke, L.Y. Chen, E. Smirni, R. Birke, L.Y. Chen, E. Smirni, Data Centers in the Wild: A Large Performance Study, Technical Report, IBM Research - Zurich, Switzerland, 2012.
[5] C. Reiss, A. Tumanov, G.R. Ganger, R.H. Katz, M.A. Kozuch, Heterogeneity and dynamicity of clouds at scale: Google trace analysis, in: Proceedings of the ACM Symposium on Cloud Computing 2012, 2012, pp. 1–18.
[6] L. Barroso, U. Hölzle, The Datacenter as a Computer an Introduction to the Design of Warehouse-Scale Machines, 24, Morgan & Claypool Publishers, 2013.
[7] Navigant Consulting Inc. SAIC, Analysis and representation of miscellaneous electric loads in NEMS, The U.S. Energy Information Administration (Navigant Reference: 160750) (2017) 1–138.
[8] X. Li, Z. Qian, S. Lu, J. Wu, Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center, Math. Comput. Model. 58 (5) (2013) 1222–1235.
[9] M. Yousif, The state of the cloud, IEEE Cloud Comput. 5 (1) (2018) 4–5.
[10] J. Kumar, A.K. Singh, Cloud datacenter workload estimation using error preventive time series forecasting models, Cluster Comput. (2019). https://link.springer.com/article/10.1007/s10586-019-03003-2#citeas.
[11] Lingyun Yang, I. Foster, J. Schopf, Homeostatic and tendency-based CPU load predictions, in: Proceedings International Parallel and Distributed Processing Symposium, IEEE Comput. Soc, pp. 1–9.
[12] S.-R. Kuang, K.-Y. Wu, B.-C. Ke, J.-H. Yeh, H.-Y. Jheng, Efficient architecture and hardware implementation of hybrid fuzzy-Kalman filter for workload prediction, Integration VLSI J. 47 (4) (2014) 408–416.
[13] X. Sun, N. Ansari, R. Wang, Optimizing resource utilization of a data center, IEEE Commun. Surv. Tutor. 18 (4) (2016) 2822–2846.
[14] S.S. Manvi, G. Krishna Shyam, Resource management for infrastructure as a service (IAAS) in cloud computing: a survey, J.Netw. Comput. Appl. 41 (2014) 424–440.
[15] R. Weingärtner, G.B. Bräscher, C.B. Westphall, Cloud resource management: a survey on forecasting and profiling models, J. Netw. Comput. Appl. 47 (2015) 99–106.
[16] J. Zhang, H. Huang, X. Wang, Resource provision algorithms in cloud computing: a survey, J. Netw. Comput. Appl. 64 (2016) 23–42.
[17] M. Amiri, L. Mohammad-Khanli, Survey on prediction models of applications for resources provisioning in cloud, J. Netw. Comput. Appl. 82 (2017) 93–113.
[18] K. Mason, M. Duggan, E. Barrett, J. Duggan, E. Howley, Predicting host CPU utilization in the cloud using evolutionary neural networks, Future Gen. Comput. Syst. 86 (2018) 162–173.
[19] J. Kumar, A.K. Singh, Workload prediction in cloud using artificial neural network and adaptive differential evolution, Future Gen. Comput. Syst. 81 (2018) 41–52.
[20] S. Kumaraswamy, M.K. Nair, Intelligent VMs prediction in cloud computing environment, in: Proceedings of the 2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon), 2017, pp. 288–294.
[21] G.M. Wamba, Y. Li, A.C. Orgerie, N. Beldiceanu, J.M. Menaud, Cloud workload prediction and generation models, in: Proceedings of the Twenty-ninth International Symposium on Computer Architecture and High Performance Computing, SBAC-PAD 2017, 2017, pp. 89–96.
[22] S. Di, C.-L. Wang, Error-Tolerant resource allocation and payment minimization for cloud system, IEEE Trans. Parallel Distrib. Syst. 24 (6) (2013) 1097–1106.
[23] H. Zhang, G. Jiang, K. Yoshihira, H. Chen, Proactive workload management in hybrid cloud computing, IEEE Trans. Netw. Serv. Manage. 11 (1) (2014) 90–100.
[24] M. Dabbagh, B. Hamdaoui, M. Guizani, A. Rayes, Energy-Efficient resource allocation and provisioning framework for cloud data centers, IEEE Trans. Netw. Serv. Manage. 12 (3) (2015) 377–391.
[25] A.K. Singh, J. Kumar, Secure and energy aware load balancing framework for cloud data centre networks, Electron Lett. 55 (2019) 540–541.
[26] F. Ramezani, A fuzzy virtual machineworkload prediction method for cloud environments, in: Proceedings of the 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2017, pp. 1–6. Naples

[27] H. Shen, L. Chen, Resource demand misalignment: an important factor to consider for reducing resource over-provisioning in cloud datacenters, IEEE/ACM Trans. Netw. 26 (3) (2018) 1207–1221.
[28] F.-H. Tseng, X. Wang, L.-D. Chou, H.-C. Chao, V.C.M. Leung, Dynamic resource prediction and allocation for cloud data center using the multiobjective genetic algorithm, IEEE Syst. J. 12 (2) (2018) 1688–1699.
[29] J. Kumar, A.K. Singh, Cloud resource demand prediction using differential evolution based learning, in: Proceedings of the 2019 Seventh International Conference on Smart Computing Communications (ICSCC), 2019, pp. 1–5.
[30] N. Liu, Z. Li, J. Xu, Z. Xu, S. Lin, Q. Qiu, J. Tang, Y. Wang, A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning, in: Proceedings of the 2017 IEEE Thirty-seventh International Conference on Distributed Computing Systems (ICDCS), IEEE, 2017, pp. 372–382.
[31] F. Qiu, B. Zhang, J. Guo, A deep learning approach for VM workload prediction in the cloud, in: Proceedings of the 2016 Seventeenth IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), IEEE, 2016, pp. 319–324.
[32] J. Kumar, R. Goomer, A.K. Singh, Long short term memory recurrent neural network (LSTM-RNN) based workload forecasting model for cloud datacenters, Procedia Comput. Sci. 125 (2018) 676–682.
[33] Q. Zhang, L.T. Yang, Z. Yan, Z. Chen, P. Li, An efficient deep learning model to predict cloud workload for industry informatics, IEEE Trans. Ind. Inf. 14 (7) (2018) 3170–3178.
[34] Y. Zhang, J. Yao, H. Guan, Intelligent cloud resource management with deep reinforcement learning, IEEE Cloud Comput. 4 (6) (2017) 60–69.
[35] Y.S. Patel, R. Misra, Performance comparison of deep VM workload prediction approaches for cloud, in: P. Pattnaik, S. Rautaray, H. Das, J. Nayak (Eds.), Progress in Computing, Analytics and Networking. Advances in Intelligent Systems and Computing, Springer, Singapore, 2018, pp. 149–160.
[36] M. Amiri, L. Mohammad-Khanli, R. Mirandola, A sequential pattern mining model for application workload prediction in cloud environment, J. Netw. Comput. Appl. 105 (2018) 21–62.
[37] M. Amiri, L. Mohammad-Khanli, R. Mirandola, An online learning model based on episode mining for workload prediction in cloud, Future Gen. Comput. Syst. 87 (2018) 83–101.
[38] S. Kim, T. Kim, C. Yoo, Workload prediction using run-length encoding for runtime processor power management, Electron Lett. 51 (22) (2015) 1759–1761.
[39] Z. Xiao, S. Member, W. Song, Q. Chen, Dynamic resource allocation using virtual machines for cloud computing environment, IEEE Trans. Parallel Distrib. Syst. 24 (6) (2013) 1107–1117.
[40] Z. Chen, Y. Zhu, Y. Di, S. Feng, Self-adaptive prediction of cloud resource demands using ensemble model and subtractive-fuzzy clustering based fuzzy neural network, Comput. Intell. Neurosci. 2015 (2015) 14.
[41] N. Singh, S. Rao, Ensemble learning for large-scale workload prediction, IEEE Trans. Emerg. Top. Comput. 2 (2) (2014) 149–165.
[42] M. Chen, J. Yuan, D. Liu, T. Li, An adaption scheduling based on dynamic weighted random forests for load demand forecasting, J. Supercomput. (2017) 1–19.
[43] I.K. Kim, W. Wang, Y. Qi, M. Humphrey, CloudInsight: utilizing a council of experts to predict future cloud application workloads, in: Proceedings of the Tenth IEEE International Conference on Cloud Computing (Cloud 2018), San Francisco, USA, 2018, pp. 1–8. July 2–July 7
[44] W. Zhong, Y. Zhuang, J. Sun, J. Gu, A load prediction model for cloud computing using PSO-based weighted wavelet support vector machine, Appl. Intell. (2018) 1–12.
[45] Y. Jiang, C. Perng, T. Li, R.N. Chang, Cloud analytics for capacity planning and instant VM provisioning, IEEE Trans. Netw. Serv. Manage. 10 (3) (2013) 312–325.
[46] C. Liu, Y. Shang, L. Duan, S. Chen, C. Liu, J. Chen, Optimizing workload category for adaptive workload prediction in service clouds, in: A. Barros, D. Grigori, N.C. Narendra, H.K. Dam (Eds.), Service-Oriented Computing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2015, pp. 87–104.
[47] K. Cetinski, M.B. Juric, Ame-wpc: advanced model for efficient workload prediction in the cloud, J. Netw. Comput. Appl. 55 (2015) 191–201.
[48] J. Kumar, A.K. Singh, Dynamic resource scaling in cloud using neural network and black hole algorithm, in: Proceedings of the 2016 Fifth International Conference on Eco-friendly Computing and Communication Systems (ICECCS), 2016, pp. 63–67.
[49] J. Kumar, A.K. Singh, An efficient machine learning approach for virtual machine resource demand prediction, Int. J. Adv. Sci. Technol. 123 (2019) 21–30.
[50] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (2006) 489–501.
[51] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proceedings of the 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541), vol. 2, IEEE, pp. 985–990.
[52] A. Hatamlou, Black hole: a new heuristic optimization approach for data clustering, Inf. Sci. (Ny) 222 (2013) 175–184. Including Special Section on New Trends in Ambient Intelligence and Bio-inspired Systems
[53] C. Reiss, J. Wilkes, J.L. Hellerstein, Google cluster-usage traces: format + schema, Technical Report, Google Inc., Mountain View, CA, USA, 2011.
[54] I. Rodríguez-Fdez, A. Canosa, M. Mucientes, A. Bugarín, STAC: a web platform for the comparison of algorithms using statistical tests, in: Proceedings of the 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2015.
[55] F.J. Baldan, S. Ramirez-Gallego, C. Bergmeir, F. Herrera, J.M. Benitez, A forecasting methodology for workload forecasting in cloud systems, IEEE Trans. Cloud Comput. 6 (4) (2018) 929–941.

Dr. Jitendra Kumar obtained his doctorate degree from National Institute of Technology Kurukshetra, India (*An Institution of National Importance*) in 2019. Currently, he is an Assistant Professor in the Department of Computer Engineering & Applications, GLA University, Mathura, India. He has more than 2 years experience in industry and academia. He has published several papers in national and international journals and conferences of high repute. His current research interests include Machine Learning, Cloud Computing, Data Analytics, Parallel Processing.

Prof. Ashutosh Kumar Singh is working as a Professor and Head in the Department of Computer Applications, National Institute of Technology Kurukshetra, India. He has more than 18 years research and teaching experience in various Universities of the India, UK, and Malaysia. He received his Ph.D. in Electronics Engineering from Indian Institute of Technology, BHU, India and Post Doc from Department of Computer Science, University of Bristol, UK. He is also Charted Engineer from UK. His research area includes Verification, Synthesis, Design and Testing of Digital Circuits, Data Science, Cloud Computing, Machine Learning, Security, Big Data. He has published more than 160 research papers in different journals, conferences and news magazines. He is the co-author of six books which includes 'Web Spam Detection Application using Neural Network', 'Digital Systems Fundamentals' and 'Computer System Organization & Architecture'. He has worked as an Editorial Board Member of International Journal of Networks and Mobile Technologies, International journal of Digital Content Technology and its Applications. Also he has shared his experience as a Guest Editor for Pertanika Journal of Science and Technology. He is involved in reviewing process of different journals and conferences such as; IEEE transaction of computer, IET, IEEE conference on ITC, ADCOM etc.

Prof. Rajkumar Buyya is a Redmond Barry Distinguished Professor and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft, a spin-off company of the University, commercializing its innovations in Cloud Computing. He served as a Future Fellow of the Australian Research Council during 2012–2016. He has authored over 625 publications and seven text books including "Mastering Cloud Computing" published by McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese and international markets, respectively. He has also edited several books including "Cloud Computing: Principles and Paradigms" (Wiley Press, USA, Feb 2011). He is one of the highly cited authors in computer science and software engineering worldwide (hindex=116, g-index=255, 70,100+ citations). Microsoft Academic Search Index ranked Dr. Buyya as #1 author in the world (2005–2016) for both field rating and citations evaluations in the area of Distributed and Parallel Computing. "A Scientometric Analysis of Cloud Computing Literature" by German scientists ranked Dr. Buyya as the World's Top-Cited (#1) Author and the World's Most-Productive (#1) Author in the Cloud Computing. Recently, Dr. Buyya is recognized as a "2016 Web of Science Highly Cited Researcher" by Thomson Reuters and a Fellow of IEEE for his outstanding contributions to Cloud computing. For further information on Dr. Buyya, please visit his cyberhome: www.buyya.com.