IET Communications

Research Article

IET Journals
The Institution of Engineering and Technology

Power consumption model based on feature selection and deep learning in cloud computing scenarios

ISSN 1751-8628
Received on 19th July 2019
Revised 9th February 2020
Accepted on 12th March 2020
E-First on 13th May 2020
doi: 10.1049/iet-com.2019.0717
www.ietdl.org

Yang Liang[1,2], Zhigang Hu[1] ✉, Keqin Li[3]

[1]*School of Computer Science and Engineering, Central South University, Changsha, People's Republic of China*
[2]*School of Informatics, Hunan University of Chinese Medicine, Changsha, People's Republic of China*
[3]*Department of Computer Science, State University of New York, New Paltz, NY, USA*
✉ *E-mail: zghu@csu.edu.cn*

Abstract: High power consumption of cloud data centres is a crucial challenge in modern cloud computing. To comply with the conceptions of green computing, power consumption prediction of the computing cluster has a major role to play in these energy conservation efforts. However, due to complexity and heterogeneity in cloud computing scenarios, it is difficult to accurately predict the power consumption using conventional approaches. To this end, this study presents a power consumption model based on feature selection and deep learning to powerfully cope with low energy efficiency. Different from other methods focusing on only a few performance attributes, the proposed method takes into account up to 12 energy-related features and introduces deep neural network architecture, aiming at making full use of massive data to train model completely. In particular, this approach is composed of three main phases including (i) performance monitoring and energy-related feature acquisition, (ii) essential feature selection, and (iii) model establishment and optimisation. Representative results of comprehensive experiments, in terms of the relative error, reveal that the proposed power consumption model can undoubtedly achieve state-of-the-art predictive capability when compared with other models in most cases.

1 Introduction

Over recent years, the scale of the data centre (DC) is increasingly expanding, with the accelerated growth of data generated by applications in various industries. Cloud computing is playing an increasingly important role in our daily lives, driving the energy consumption (EC) of DC to new highs. The energy management issue of DC has drawn people's attention extensively [1, 2]. The industry generally uses power usage effectiveness (PUE) to measure the energy efficiency of DC. As an effective engineering ratio, PUE is shown in the following equation:

$$\mathrm{PUE} = \frac{\sum (P_m + P_e + P_{\mathrm{IT}} + P_{\mathrm{other}})}{P_{\mathrm{IT}}} \quad (1)$$

where the numerator denotes the power consumption of all facilities in the DC including mechanical devices, electrical equipments, IT infrastructures, and other necessary devices, while the denominator denotes the power consumption of IT infrastructures, providing real useful IT work. The closer the value of PUE is to 1, the higher the energy efficiency in DC is.

According to several DC industry surveys, provided by Uptime Institute as well as other professional consulting institutions, the average power usage effectiveness (APUE) has only improved from 2.5 in 2007 to 1.89 in 2011 while it only improved from 1.89 to 1.7 in 2014 [3]. In the USA, the APUE is still around 1.7 in 2017 for enterprise DCs, while it is between 1.5 and 1.6 for the newly built DCs. Very few advanced DCs, such as those of Facebook and Google, can achieve PUE of 1.2 or lower. However, it is obvious that the improvements of energy efficiency measured by PUE are slowing down significantly. Even as excellent as Google, it is getting harder and harder to further reduce the PUE value. As after a certain degree, the interaction between refrigeration and electrical systems and complex feedback loops make it difficult to accurately use conventional engineering formulas to derive and optimise DC efficiency [4, 5]. Thus, developing innovative methods or strategies for reducing PUE, that is minimising EC and operational cost eventually, deserves our endeavours.

Over the last decade, energy efficiency is a major research concern in modern cloud computing [6]. Many works, some of which will be discussed in Section 2, provide various solutions to estimate power consumption of a dynamic and complex DC. Some researchers have employed machine learning (ML) techniques, driven by data essentially, to resource provision and management in cloud DC for ensuring energy efficiency [7–10]. ML is based on training data to build statistical models, helping us to make decisions or predictions about events in the real world. There are many traditional models including support vector machines, decision tree, KNN, naive Bayes, K-means, neural networks (NNs), and so on, which belong to ML. In this paper, we present an improved deep learning (DL) framework, driven by massive data, to offer a solution for improving energy efficiency by predicting next energy demand in the cloud. Considering the non-linearity and complexity of various devices in DC and accuracy of the experiment [11], we focus on researching the power consumption of IT infrastructures, which is expressed as P_{IT} in (1). To this end, our goal is to improve the energy efficiency of cloud computing clusters by allocating resources legitimately using our power consumption model.

Although modelling and predicting real-world dynamic system behaviours have received widespread research interest, approximating any non-linear or complex system remains a challenging task. To address these challenges, the significant contributions of this work can be summarised as follows:

i) A deep NN architecture is elaborated and a power consumption model based on feature selection and DL framework is proposed, achieving a boost in prediction accuracy regardless of the application belonging to I/O intensive, compute-intensive or transactional web.
ii) An elaborate-designed data preprocessing process including performance monitoring, energy-related feature acquisition, and key feature selection has been provided, taking into account more comprehensive energy-related features and removing redundant features effectively based on information theory.

iii) Extensive and multi-angle experimental analysis are implemented for training and validating the mathematical model, as well as the research domain accredited benchmarks are used for data collection and feature extraction by performing tasks of different application scenarios.
iv) Analyses and comparisons between six different power consumption models in terms of the relative error (RE) demonstrate the proposed model is effective and vastly superior.

The rest of the paper is organised as follows. Cloud computing energy efficiency related works are reviewed in Section 2. In Section 3 and 4, the well-designed modelling processes including data preprocessing and model constructing are described in detail. The analysis and comparison of experimental results is presented in Section 6. Finally, Section 7 summarises the paper with recommendations for future work in this area.

2 Related work

No doubt that energy efficiency in cloud computing has become a hot theme in recent years [12]. Many solutions have been carried out for modelling the aggregate power consumption of a server. As servers conduct most of the work in DCs, they are the most power proportional components in a DC. Roy *et al.* [13] present a server power model as a summation of CPU and memory power consumption, which is one of the simplest power models. Their power model was represented as

$$E(A) = E_{\text{cpu}}(A) + E_{\text{memory}}(A) \tag{2}$$

where $E_{\text{cpu}}(A)$ and $E_{\text{memory}}(A)$ are EC of the CPU and the memory while running Algorithm 1.

Considering more components of a server, Tudor *et al.* [14] propose a power model expressed as a function of energy used by CPU, memory, and I/O devices. Furthermore, Song *et al.* [15] describe a similar power model by expressing the system power consumption as a summation of CPU, memory, disk, and network interface card, which can be shown as

$$E_{\text{total}} = E_{\text{cpu}} + E_{\text{memory}} + E_{\text{disk}} + E_{\text{NIC}} \tag{3}$$

Another energy model can be further constructed considering the levels of resource utilisation by the key components of a server as [16]

$$P_t = C_{\text{cpu}}U_{\text{cpu}} + C_{\text{memory}}U_{\text{memory}} + C_{\text{disk}}U_{\text{disk}} \tag{4}$$

where U_{cpu} is the CPU utilisation, U_{memory} is the memory access rate, U_{disk} is the hard disk I/O request rate, and U_{NIC} is the network I/O request rate. P_t refers to the predicted power consumption of server at time t while C_{cpu}, C_{memory}, C_{disk}, and C_{NIC} are the coefficients of CPU, memory, disk, and NIC, respectively.

Tian *et al.* describe the power consumption of a server as follows [17]:

$$P_i = u_i k_i \mu_i^{\alpha_i} + P_i^* \tag{5}$$

where $\mu_i^{\alpha_i}$ denotes the service rate and u_i represents the utilisation of server i. P_i^* is the static power consumption of server i. k_i, α_i, P_i^* are constants determined by the DC.

A different version of model is created by Mills *et al.* to estimate the system EC [18]. The power consumed by a server in this power model is given by

$$E(\sigma, [t1, t2]) = \int_{t=t1}^{t2} \left(\sigma^3 + \rho\sigma_{\max}^3\right) \mathrm{d}t \tag{6}$$

where ρ stands for overhead power which is a fixed factor of the power consumed when the CPU is operating at full speed. σ denotes the execution speed of the processor. The overhead includes the power consumption by all other system components such as memory, network, and so on.

What is more, in recent years, numerous studies of improving energy efficiency in DCs have been performed [19–22]. In terms of the results, most researchers achieve up to a large proportion of reduction in energy usage. However, the studies for predicting power consumption is insufficient while power consumption estimation is critical to resource provisioning and task scheduling [23–25].

According to different application scenarios, Zhou *et al.* [26] describe the mathematical expressions of the power regression model in detail and verify that the power regression model has higher precision than other regression models (linear regression, exponential regression, and polynomial regression).

Zhang *et al.* [27] find that the cubic polynomial model could obtain better results whether EC swing in a small range or a wide range, exceeding the linear model that is only suitable for a small range.

Bertran *et al.* [28] prove that the performance monitoring counters (PMC) power model still has good performance, whose average error is less than 3%.

Park and Mun [29] propose a prediction approach that power consumption is determined by basic power and active power, not depending on actual power consumption. The basic power is constant determined by hardware while the active power can be easily calculated with the help of Cloud Monitor. The results show that average error rate is about 4.22% in the CPU test.

Liu *et al.* [30] present an abstract energy consumption (AEC) model to estimate EC of the cloud computing. Particularly, considering the source-level EC of a task is not the simple summation of that of its statements, two quantitative measurements, 'cross-degree' and 'reuse-degree', are proposed as the code structure features. Experimental results show that the mean deviation between the EC and AEC is around 0.005.

Yu *et al.* [31] construct a CMP EC model by only choosing some dominant parameters such as CPU, memory, disk, and so on. Moreover, taking into account different scenarios, they mainly focus on EC changes of computer systems in data-intensive and compute-intensive states. Experiments show that the CMP model can achieve a higher accuracy on power estimation than FAN and Cubic mode, especially in data-intensive scenario.

Foo *et al.* [32–34] develop an EC prediction model based on an evolutionary NN combining with several novel mechanisms of a genetic algorithm. The results, both in terms of forecasting speed and accuracy, suggest that the evolutionary NN approach to EC forecasting for cloud computing is highly promising.

As ensuring energy efficiency in DCs is a worthwhile endeavour, more and more innovative approaches, such as machine learning, are applied to resource conservation and management problems in the cloud computing [9].

Given the complexity of distributed clusters and the interaction among multiple control systems [35], accurate energy efficiency models allow DC managers to optimise operational configurations without on-site commissioning.

Inspired by the challenges and deficiency of the existing works, we present a power consumption model based on feature selection and deep learning (abbreviated as FSDL model) in cloud computing scenarios. While drawing on this idea from aforementioned efforts, our goal is quite different because we seek not only energy efficiency but also the high prediction accuracy of power consumption, leading to novel methodology. Section 3 and Section 4 will elaborate the procedures of our approach. Section 5 reveals the comprehensive experimental results.

3 Methodology

The methodology is mainly composed of the following stages: performance monitoring, energy-related feature acquisition, feature selection, building a power consumption model, and evaluation of experimental results. Fig. 1 shows each step of constructing a power consumption model by the methodology proposed in this paper.

The performance monitoring step is responsible for monitoring all relevant performance metrics as the primary data source. Then energy-related features are obtained and formatted. At the stage of essential feature selection, we calculate the importance ratio of each feature based on information entropy theory so that core features can be selected. Based on above, we build a power consumption model based on DL approach and optimise the parameters. Finally, we evaluate the results of the experiment and demonstrate the effectiveness of the proposed model.

4 Data preprocessing

As one type of ML approaches, DL is evolved from the deep research in NNs while its network structure is more complex than other ML models. DL is used to learn the essential characteristics and laws of large amount of sample data entered. Therefore, the quality and quantity of sample data play a crucial role in training the DL models. In this work, we first designed the process of data preprocessing, specifically including energy-related feature acquisition and essential feature selection, to ensure the quality of the training data set.

4.1 Performance monitoring and energy-related feature acquisition

Data acquisition is an important part of performance monitoring and analysing in cloud computing, and it is the foundation of data processing, analysis, and display. Moreover, as the MapReduce parallel computing framework is a widely used programming model in DCs, it has a significant impact on the EC of the DC. Hence, experimental data sets are collected from the Hadoop cluster while the MapReduce jobs are running.

To obtain the working status of the servers and collect experimental data in time, it is extremely significant to monitor and manage the performance metrics of the server cluster. To the best of our knowledge, many commonly used performance monitors, i.e. Ganglia, Hadoop build-in counters, Nagios, Nigel's performance Monitor (nmon), Cloudera Manager, Zenoss, Zabbix, and so on, can contribute to data monitoring and acquisition. For instance, Ganglia, a scalable distributed monitoring system for high-performance computing systems, is mainly used to collect system-level information including CPU utilisation, disk utilisation, and so on. Nagios, an open-source application, can monitor systems, networks, and infrastructure. Besides offering host monitoring, Nagios can provide an anomaly detection mechanism when things go wrong. Zabbix, an open-source monitoring tool for networks, operating systems, and applications, can monitor statistics such as CPU load, network utilisation, disk space, and so on. However, most of them can only monitor some basic performance metrics and cause excessive system overhead.

Considering the limitations and defects of a separate performance monitoring tool in cloud computing platforms, our inspiration was to design a comprehensive solution for energy-related features acquisition, combining several typical and complementary open-source solutions. To achieve a trade-off between the system overhead and the scope of monitoring metrics, we proposed a combined monitoring method consisting of Ganglia, Hadoop build-in counters, and Zabbix in this work. The main advantages of these components are as follows: Ganglia can monitor basic performance metrics of a cluster in real time, with low system overhead and no impact on the performance of related services; Hadoop build-in counters are mainly used to report multiple indicators related to MapReduce jobs and the results do not need to be transmitted over the network; Zabbix supports secondary custom development to monitor various required performance parameters.

This combined monitoring approach effectively fused the strengths of these three components, not only co-monitoring more diverse energy-related metrics but also ensuring a lower overall system overhead. The required energy-related metrics and the corresponding monitoring ways are shown in Table 1.

4.2 Essential feature selection

Theoretically speaking, all energy-related features need to be monitored and recorded. Nevertheless, there are huge differences between different attributes for the importance of decision making, even affecting the correctness of decision. During data acquisition and performance monitoring, data sets usually have problems such as irrelevant attributes and redundant attributes [36]. To solve the problem of effective information redundancy in high-dimensional data, we will discuss some classical data dimensionality reduction algorithms and the process used to select a subset of relevant features from those extracted in the preceding section.

As one of the most widely used unsupervised dimensionality reduction algorithms, principal component analysis (PCA) [37] obtains the projection matrix by maximising the variance among samples after dimensionality reduction, so as to preserve the distribution properties of global samples as much as possible. However, the PCA algorithm consumes a large amount of memory and has a high time complexity. As a result, it is difficult to calculate the eigenvectors of high-dimensional data, and the processing time cannot meet the requirements of related applications.

Linear discriminant analysis (LDA) [38], a supervised linear dimensionality reduction algorithm, can obtain a projection subspace with strong discriminative ability by maximising the quotient between inter-class dispersion and intra-class dispersion to separate the samples of different classes as far as possible. However, LDA focuses on the larger inter-class dispersion and ignores the smaller inter-class dispersion, which is likely to cause the fusion of different classes that are close to each other in the subspace, resulting in excessive reduction and information loss of the subset.

Different from PCA and LDA, locally linear embedding (LLE) [39] belongs to the category of non-linear dimensionality reduction methods. Under the premise of keeping the original data properties unchanged, LLE can leverage local linearity to represent global non-linearity and map data from high-dimensional space to low-dimensional space to determine the subset features. However, LLE is sensitive to the selection of the nearest-neighbour number, as different nearest-neighbour numbers have a great influence on the dimensionality reduction results in the high-dimensional space, which will lead to a large fluctuation in the feature selection results.

Therefore, considering the problems existing in the traditional data dimensionality reduction algorithm above, we have to develop a more effective, targeted attribute reduction mechanism to optimise the feature set of Table 1. Concretely, we will calculate the importance ratio of each feature, based on information entropy

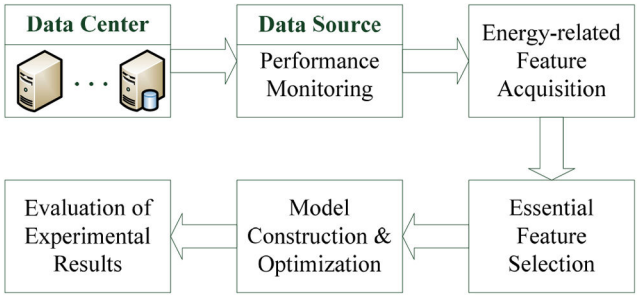

Fig. 1 *Overall framework of model construction*

Table 1 Energy-related metrics and corresponding monitoring ways

Number	Metrics	Unit	Description	Acquisition method
1	CPU utilisation	percent	percent of time that the CPU is utilised	Ganglia
2	average load	loads/procs	average number of processes simultaneously in ready state during the last minute	Ganglia
3	memory use	gigabytes	memory use for processes	Ganglia
4	map task read	gigabytes	data read by map task from local disk	Hadoop built-in counters
5	reduce task read	gigabyte	data read by reduce task from local disk	Hadoop built-in counters
6	map task write	gigabyte	data written by map task to local disk	Hadoop built-in counters
7	reduce task write	gigabyte	data written by reduce task to local disk	Hadoop built-in counters
8	shuffle size	gigabyte	data transferred from map to reduce	Hadoop built-in counters
9	network I/O speed	gigabytes/s	data transmitted and received	Ganglia
10	file size	gigabytes	size of MapReduce jobs	Hadoop built-in counters
11	number of M/R instructions	number	instruction number of jobs	Zabbix
12	average jobs completion duration	hour	time taken to finish a MapReduce job	Hadoop built-in counters
13	disk utilisation	percent	percent of time that the disk is utilised	Ganglia
14	transmission and read/write ratio	percent	data transfer and read/write ratio	Zabbix
15	available space in file system	gigabytes	file system free space	Zabbix
16	page faults/s	number	speed of page errors caused by executing threads in the process	Zabbix
17	bytes consumed per CPU second	bytes	average number of bytes consumed per CPU second	Zabbix
18	context switches rate	number	number of switches between processes or threads per second	Zabbix
19	HDFS R/W throughput	bytes/s	amount of data successfully transmitted per second in HDFS	Hadoop built-in counters
20	disk traffic	bytes/s	rate of disk transfers	Zabbix
21	paging rate	number	number of pages READ to physical memory or written to pagefile(s) per second	Zabbix
22	power consumption	kw	power consumed by Hadoop cluster	smart meter

of rough set theory, to remove redundant attributes to streamline the dimension of the feature vector.

Definition 1: (Feature selection): Select $M(M<N)$ sub-features from N features. Among the M sub-features, the objective function can achieve the optimal solution.

Selecting as few sub-features as possible will not significantly reduce the effect of the model. Conversely, building a model by selecting partial core sub-features can greatly reduce the execution time of the learning algorithm and improve the efficiency of the model.

Information entropy is the average rate at which information is produced by a stochastic source of data [40]. Information entropy is a measure of the degree of confusion or dispersion of distribution. The more dispersed the distribution (or the more even the distribution) is, the larger the entropy gets. By contrast, the more orderly the distribution (or the more concentrated the distribution) is, the smaller the entropy becomes.

Definition 2: (Information entropy): Let category be X and the corresponding label be x, $x \in X$. Let $P(x)$ denotes the probability of x; then the information entropy of X is defined as

$$H(x) = -\sum_{\forall x \in X} P(x)\log_b P(x) \tag{7}$$

where b is the base of the logarithm used. In (7), b can be set as 2 as the entropy is typically measured in bits. The measure of information entropy associated with each possible data value is the negative logarithm of the probability mass function for the value.

In information theory, the conditional entropy quantifies the amount of information needed to describe the outcome of a random variable Y given that the value of another random variable X is known. The entropy of Y conditioned on X is written as $H(Y|X)$.

Definition 3: (Conditional entropy): Let x be the value of attribute X and let y be the value of attribute Y. If $H(Y|X=x)$ is the entropy of the discrete random variable Y conditioned on the discrete random variable X taking a certain value x, then $H(Y|X)$ is the result of averaging $H(Y|X=x)$ over all possible values x that X may take. Then the conditional entropy can explicitly be written as

$$\begin{aligned} H(Y|X) = & -\sum_{\forall x \in X} P(X) \sum_{\forall y \in Y} P(y|x)\log_b P(y|x) \\ & \sum_x P(x)H(Y|X=x) \end{aligned} \tag{8}$$

Definition 4: (Kullback–Leibler divergence): the Kullback–Leibler divergence (KLIC) is a measure of how one probability distribution diverges from a second, expected probability distribution. Then the KLIC of attribute A is defined as the difference between the basic entropy and the conditional entropy of its attribute

$$D_{\mathrm{KL}}(A) = H(S) - H(S|A) \tag{9}$$

where $H(S)$ is the basic entropy, while $H(S|A)$ is the conditional entropy under the condition of attribute A. The greater the attribute's KLIC is, the more information the attribute can provide. Thus, KLIC describes the purity of an attribute.

According to (7) and (8), assume that C is the collection of load performance attributes collected from the operating system, and D is the cluster power consumption value when each feature value is acquired during the execution of jobs. Then the conditional entropy of each performance attribute can be expressed as

$$H(D|C) = -\sum_{i=1}^{n} P(X_i) \sum_{j=1}^{m} P(Y_j|X_i)\log_b P(Y_j|X_i) \tag{10}$$

In cloud computing, various performance attributes have different effects on power consumption of computing clusters. Our goal is to apply information theory to detect the 'weightier' input features that contribute positively to the power consumption. From this perspective, assume that C_i is an arbitrary attribute in the conditional attribute set C, then its importance relative to the system power consumption can be described as the entropy change of C relative to decision attributes before and after C_i is removed from C. Then the importance of C_i is quantified as

$$D_{\mathrm{KL}}(C_i) = H(D|C - \{C_i\}) - H(D|C) \tag{11}$$

Here, we can achieve the purpose of attribute reduction by calculating the importance ratio of each attribute in C according to the above equations.

5 Power consumption model implementation

DL approach can effectively model and predict non-linear dynamic systems such as the cloud DCs, which is a viable solution for reducing power consumption in cloud computing.

5.1 Deep neural network architecture (DNNA)

DL allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction [41]. As a multilayered complex NN structure, the DL model can effectively solve the problem that the shallow network usually cannot converge to the global optimal solution.

The DNNA, a framework used for power consumption modelling, consists of three main components: the input layer, the hidden layer, and the output layer. The input layer contains d input nodes, depending on the dimension of the input feature vector, and 1 bias node, whose common value is 1. The hidden layer usually has m layers and each layer has k hidden nodes and 1 bias node, while the most appropriate values of m or k depend on the comparison of multiple experiments. The output layer has only 1 output node representing the power consumption. Fig. 1 illustrates the structure of DNNA.

In the DNNA we specify a set of n vectors $\boldsymbol{FV}_1, \boldsymbol{FV}_2, \ldots, \boldsymbol{FV}_n$, where each vector $\boldsymbol{FV}_i \in R^d$ (d-dimensional of feature vector space). EPC, an abbreviation of electrical power consumption, denotes the result estimated by the entire DNNA at the output layer.

In a forecasting process using DNNA, specifically, the input matrix X, consisting of w feature vectors, is multiplied by the model parameters matrix $\boldsymbol{\theta}$ to produce the first hidden state matrix $\boldsymbol{H}_1$. The next hidden layers continue to forward propagate intermediate results using the same computational process until the predicted power consumption value is calculated.

5.2 Mathematical model

Fig. 2 presents the elegant design of the DL architecture which can search for relationships between complex input features and power consumption by training massive amounts of data. Understanding the underlying mathematical behaviour of DNNA allows us to control and optimise it.

Let the DNNA input layer comprise of d input eigenvalues denoted as $(F_1, F_2, \ldots, F_d)$. Starting from the second layer, specially, a_j is the input of each neuron that can be generated by

$$a_j = \sum_{i=1}^{n} F_i \times W_{ij}^{1} + 1 \tag{12}$$

where W_{ij}^{1} is the connection weight between the input layer and the first layer of the hidden layers. i identifies the neuron of the preceding layer while j identifies the neuron of the current layer. a_j is the weighted sum of the outputs of all neurons in the preceding layer.

Then the output of the jth neuron is determined by an activation function shown in (14) acting on it

$$\varphi(z) = \frac{1}{1 + e^{-z}} \tag{13}$$

Equation (13) is the sigmoid function usually used as an activation function. The argument z will receive an assignment from $\delta_j(r)$, the input variable of the jth downstream neuron.

Assume there are m layers in the hidden layer. Considering the applicability of the neurons in each layer, a more general mathematical expression based on (12) is induced as follows:

$$\delta_j(r) = \sum_{i=1}^{k} \varphi(\delta_i(r-1)) \times W_{ij}^{r-1} + 1 \tag{14}$$

where $\delta_j(r)$ denotes the input of the jth neuron of the rth layer. r should belong to the scope of $2 \le r \le (m+2)$. k is the number of neurons of the $(r-1)th$ layer.

Let EPC be the predicted results of DL NNs at the output layer. More generally, the power consumption prediction model based on (14) can be deduced as

$$\mathrm{EPC} = \sum_{i=1}^{k} \varphi(\delta_i(m+1)) \times W_{i1}^{m+1} + 1 \tag{15}$$

During the process of DL, however, the model needs to continuously learn a large amount of training samples in order to determine the optimal parameters of the model. The evaluation criterion for the final solution depends on whether the cost function has approximated the minimum value and hence the cost function is given by

$$M(\theta) = \frac{1}{2s} \sum_{i=1}^{s} \left(\hat{Y}^{(i)} - Y_\theta\left(x^{(i)}\right) \right)^2 \tag{16}$$

where $\hat{Y}^{(i)}$ is the target at the output layer while $Y_\theta\left(x^{(i)}\right)$ is the actual calculated value by the DL model. s is the number of samples. $M(\theta)$ denotes the error between the fitted value and the true value. The partial derivative of $M(\theta)$ to θ would be

$$\frac{\partial M(\theta)}{\partial \theta_j} = -\frac{1}{s} \sum_{i}^{s} \left(\hat{Y}^{(i)} - Y_\theta\left(x^{(i)}\right) \right) x_j^{(i)} \tag{17}$$

Essentially speaking, DL discovers intricate structure in large data sets by using the back-propagation (BP) algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Thus, the choice of BP algorithm is crucial for faster convergence and higher accuracy. Gradient descent algorithm, one of the most common optimisation methods used in machine learning, can adjust the model parameters at each iteration along the opposite direction of the gradient of the cost function until a solution tends to converge on the global optima.

Gradient descent algorithm mainly contains three different forms: batch gradient descent, stochastic gradient descent, and mini-batch gradient descent (MBGD). Comprehensively considering the convergence speed and the accuracy of the optimal solution, we adopt MBGD, a compromised algorithm, to optimise the model parameters. According to (16) and (17), the pseudo-code of MBGD is described in Algorithm 1. s is the total length of the training sample set, b is the size for a mini-batch samples, and α is the learning rate.

Algorithm 1: Pseudo-code of MBGD
1: $s \leftarrow N, b \leftarrow B, \alpha \leftarrow A, n \leftarrow N/B$
2: **for** $j = 0$ to n **do**
3: **for** $i = 1 + b*j$ to $i = 1 + b*(n-1)$ **do**
4: $\theta_j = \theta_j + \alpha * \frac{1}{b} \sum_{k=i}^{i+b-1} \left(\hat{Y}^{(k)} - Y_\theta\left(x^{(k)}\right) \right) x_j^{(k)}$
5: **end for**
6: **if** $M(\theta_j) \le \varepsilon$ **then**
7: $\theta = \theta_j$
8: **break**
9: **end if**
10: **end for**
11: **return** θ

At each iteration, MBGD divides the training set into several small batches, which updates parameters with a fixed number of samples by calculating the derivatives of relevant parameters and the cost function value.

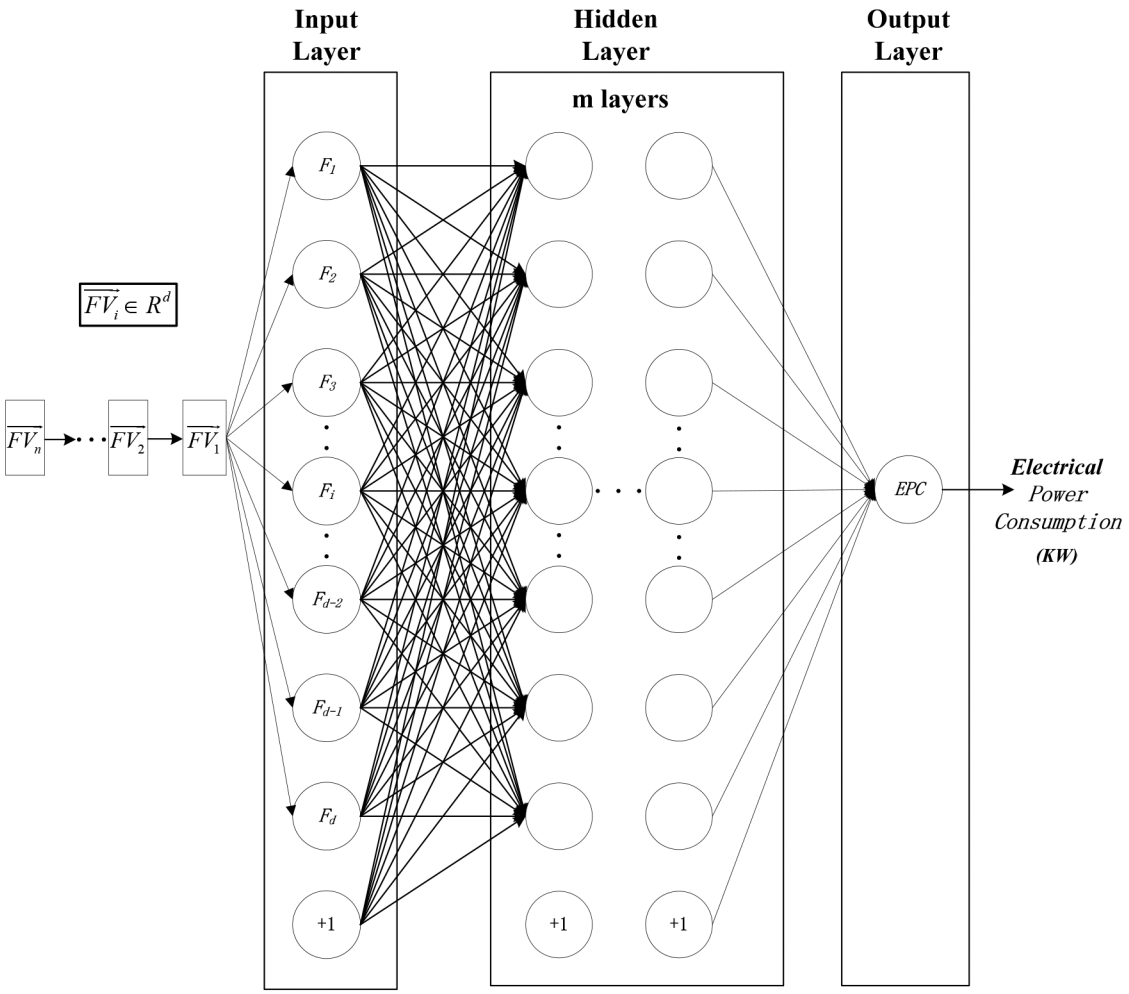

Fig. 2 *Deep neural network architecture*

6 Analysis and comparison of experimental results

In this section, we design a set of comprehensive experiments to evaluate the performance of the proposed prediction model. The Hadoop framework is adopted to simulate a cloud environment.

6.1 Experimental settings

To perform the experiments in a real environment, we use a heterogeneous cluster of ten nodes comprising one Core i5-based namenode and nine Core i3-based datanodes. After energy related metrics are extracted from the running MapReduce jobs, we use these preprocessed data to train and calibrate the prediction model on another PC. The detailed configuration is specified in Table 2.

As MapReduce jobs are generally divided into three categories: I/O intensive applications, compute-intensive applications, and transactional web applications, we adopted benchmarks released by Purdue University [42] to train the power consumption model repeatedly as well as verify its validity. The benchmark characteristics are listed in Table 3.

6.2 Feature importance

In our experiment, different categories of assignments are submitted to the Hadoop cluster. Performance features given by Table 1 are collected every 10 s during job execution. Meanwhile, we record the cluster power consumption in the sampling interval. Then we combine the feature group and the corresponding power consumption to generate an original sample.

In order to screen out the features having greater impact on power consumption, we calculate the KLIC, defined in (9), of each feature. The importance of all monitored energy-related features is shown in Table 4.

As presented in Table 4, we calculate the importance ratio and average importance ratio of all energy-related features. Even the same feature may have different contributions to power consumption in different application domains. For instance, 'CPU utilisation' is very crucial for compute-intensive applications while its importance ratio much lower than 'disk traffic' for I/O intensive applications. However, for the sake of being possessed of stronger applicability and generality, we will focus on the average importance indicators in consideration of various application scenarios.

6.3 Analysis and comparison

Based on the previous data preparation, we collect 50,000 feature vector samples. Then we divide the sample set into two parts for different stages of modelling. In the training stage, the training set with a scale of 40000 samples is used for training the model, while the test set containing 10,000 samples is responsible for model verification in the verification stage.

Furthermore, without loss of generality and for the ease of analogy with the aforementioned methods, we define the RE of power consumption in (18) as a metric to evaluate the accuracy of all models

$$\mathrm{RE} = \left| \frac{\mathrm{Power}_{\mathrm{real}} - \mathrm{Power}_{\mathrm{predict}}}{\mathrm{Power}_{\mathrm{real}}} \right| \tag{18}$$

where RE denotes the relative error, $\mathrm{Power}_{\mathrm{real}}$ is the actual power consumption value while $\mathrm{Power}_{\mathrm{predict}}$ is the calculated value from the model.

At the beginning of the modelling experiment, we have been trying to train models with input eigenvectors of different dimensions determined by different importance ratios. Moreover,

the weights in each layer were initialised from a zero-mean Gaussian distribution with standard deviation 0.01. The mini-batch size of MBGD is 100. The learning rate is set to 0.04 at the outset and is adaptively reduced by ten times when the error plateaus. In order to seek the best importance threshold, the contrast for different thresholds in terms of calculation time and RE (prediction accuracy) of the model is shown in Fig. 3.

As presented in Fig. 3, we compare the calculation time and prediction accuracy of the model under different thresholds. The magnitude relationship between the average importance ratio of a feature and a certain threshold will determine if the feature can be used as an input to the model under the current condition. We can see that as the threshold increases, the calculation time decreases, while the error decreases first and then rises. The decrease in computation time is due to the decrease in input features. The more input features do not necessarily lead to smaller errors; on the contrary, it often increases occurrence probability of overfitting. For example, when the importance threshold is 0.020, the dimension of feature vector is 10 and the RE is 0.029; when the importance threshold is 0.010, the dimension of feature vector is 12 and the RE is 0.012; when the importance threshold is 0.007, the dimension of feature vector is 15 while the RE is 0.015. The reason includes two folds. On the one hand, the parameter dimension will not be enough to fully train the model if the threshold is too large,

Table 2 Experimental environment

Item	Description
namenode	Dell OptiPlex 7040, 4*CPU Intel Core i5-6500, Memory 8 GB DDR4, Disk 1TB 7200 rpm, OS CentOS V6.4, Apache Hadoop V2.7.1
datanode	Dell Vostro 3470-R1328R, 4*CPU Intel Core i3-7100, Memory 4 GB DDR4, Disk 1TB 7200 rpm, OS CentOS V6.4, Apache Hadoop V2.7.1
pc for building model	Dell OptiPlex 7040, 4*CPU Intel Core i5-6500, Memory 4 GB DDR4, Disk 2*1TB 7200 rpm,GPU NVIDIA GTX1070, OS Windows 10, CUDA V8.0.6, Anaconda3V42.0, TensorFlow-GPU V1.3
smart meter	VICTOR VC470, rated voltage 220 V, Rated current 10 A
monitor software	Ganglia, Hadoop built-in counters, Zabbix
IDE	Eclipse V4.5.2, PyCharm V 2018.2.1

Table 3 Benchmark characteristics

Benchmark	Input size, GB	Input data	Category
self-join	250	synthetic	I/O intensive
tera-sort	300	synthetic, random	I/O intensive
ranked-inverted-index	205	multi-wordcount output	I/O intensive
kmeans	100	100 Netflix data, k = 6	compute-intensive
inverted-index	250	Wikipedia	compute-intensive
term-vector	250	Wikipedia	compute-intensive
word-count	250	Wikipedia	compute-intensive
multi-word-count	250	Wikipedia	transactional web
histogram-movies	215	Netflix data	transactional web
histogram-ratings	215	Netflix data	transactional web
grep	250	Wikipedia	transactional web

Table 4 Importance ratio of all monitored features

Number	Features	Importance ratio			Average importance ratio
		I/O intensive	Compute-intensive	Transactional web	
1	CPU utilisation	0.0364	0.0566	0.0528	0.0486
2	bytes consumed per CPU second	0.0317	0.0517	0.0442	0.0425
3	context switches rate	0.0370	0.0553	0.0419	0.0447
4	number of M/R instructions	0.0544	0.0510	0.0600	0.0551
5	shuffle size	0.0598	0.0321	0.0412	0.0444
6	disk traffic	0.0628	0.0386	0.0469	0.0494
7	HDFS R/W Throughput	0.0613	0.0310	0.0517	0.0480
8	transmission and read/write ratio	0.0511	0.0234	0.0446	0.0397
9	average load	0.0199	0.0291	0.0239	0.0243
10	memory use	0.0207	0.0188	0.0241	0.0212
11	page faults/sec	0.0135	0.0174	0.0182	0.0164
12	network I/O speed	0.0149	0.0136	0.0233	0.0173
13	paging rate	0.0162	0.0057	0.0071	0.0097
14	disk utilisation	0.0124	0.0038	0.0084	0.0082
15	available space in file system	0.0118	0.0045	0.0069	0.0077
16	MapTask read	0.0131	0.0024	0.0040	0.0065
17	reduce task read	0.0126	0.0017	0.0038	0.0060
18	map task write	0.0143	0.0020	0.0044	0.0069
19	reduce task write	0.0139	0.0028	0.0036	0.0068
20	file size	0.0035	0.0019	0.0021	0.0025
21	average jobs completion duration	0.0030	0.0028	0.0033	0.0030

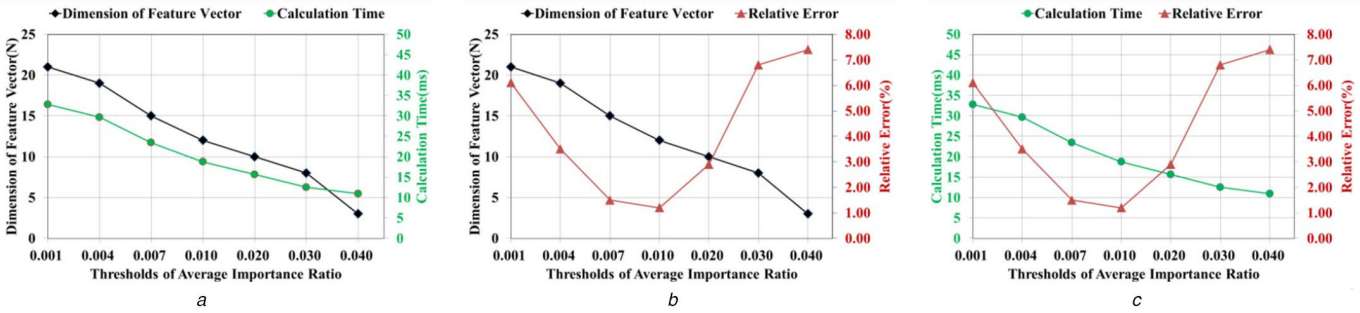

Fig. 3 *Experimental comparison of different thresholds*

(a) Variation of eigenvector dimension and calculation time under different threshold conditions, ***(b)*** Variation of eigenvector dimension and relative error under different threshold conditions, ***(c)*** Variation of calculation time and relative error under different threshold conditions

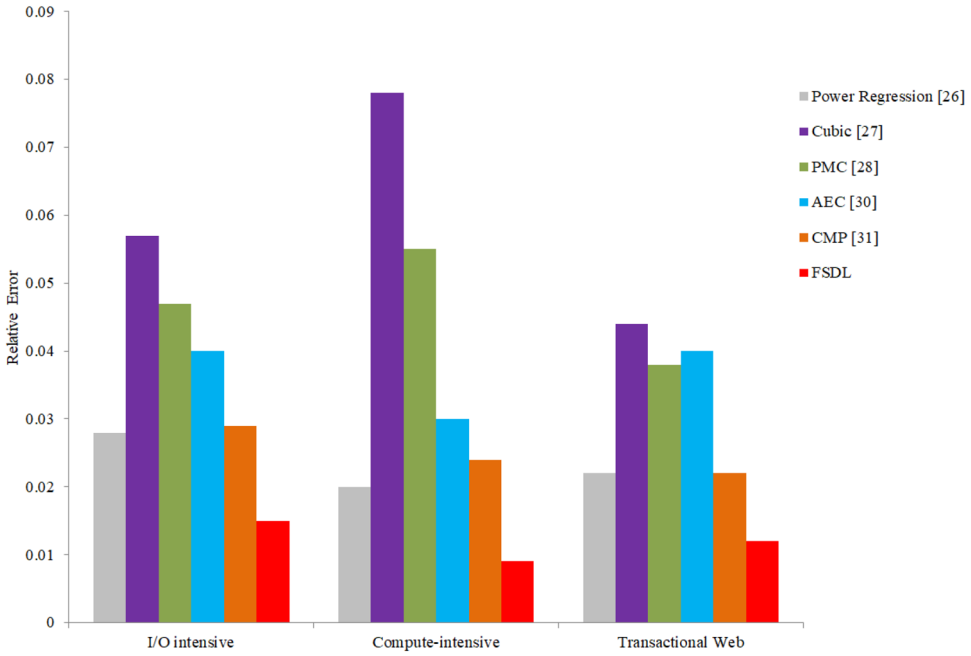

Fig. 4 *Comparison of relative errors of different models*

causing under-fitting results. On the other hand, considering too many factors to build the model if the threshold is too small will lead to over-fitting results.

Consequently, considering the tradeoff between computation time and prediction accuracy, we adopt 0.010 as the importance threshold. Then features whose average importance ratios are greater than the threshold are considered. Finally, 12 energy-related features are selected, constituting a feature vector as the input of the model.

Besides, through constantly training and comparing different NN structures, we finally adopt a five-layer network structure of $12 \times 100 \times 100 \times 100 \times 1$ for the sake of model accuracy and calculation speed.

In the verification stage, each feature vector in test set is inputted into the trained model, and thus, the power consumption of this vector is predicted. To evaluate the efficiency of FSDL model, we compare our approach with other five methods including power regression model [26], cubic model [27], PMC model [28], AEC model [30], CMP model [31]. Fig. 4 illustrates the REs of the six energy models under the same experimental environment and benchmarks.

Fig. 4, respectively, show the comparison for the six energy models in terms of RE. Fig. 4 reveals that, no matter in any application domains, the power consumption model based on FSDL can achieve the highest prediction accuracy in most cases. Compared with the other five models, the average RE of the FSDL model is reduced by more than 1.1%. Considering more core features and using deep learning method to construct the power consumption model can account for this fact. Compared with power regression model, FSDL model achieves around 1% accuracy improvement. We can see that FSDL model selects 12 energy-related features to train the model while power regression model only considers six factors (processor time, disk byte/s, disk time, page fault/s, memory used, and byte total/s). Generally, more input features may cause longer computation time and more power consumption. Compared with power regression model, although FSDL model achieves only around 1% accuracy improvement, the energy saved by model prediction will be much greater than the energy consumed by itself, especially for large-scale DCs. Therefore, it is worthwhile constructing a power consumption model based on feature selection and deep learning.

7 Conclusion and future work

In this work, we motivate, develop, and evaluate a power consumption model based on feature selection and deep learning in cloud computing. The proposed methodology is mainly composed of the following three processes: (i) performance monitoring and feature extraction, this step is responsible for obtaining all energy-related features; (ii) feature selection, this step is responsible for reserving really significant features while removing redundant features; (iii) modelling and evaluation, the last but not least step is responsible for designing, building, and evaluating model. Considering more energy-related features, the proposed solution conduces to make full use of massive sample data to train model completely with deep learning method to achieve higher prediction accuracy. In contrast to the models based on conventional approaches, the proposed power consumption model based on feature selection and deep learning has a smaller RE of power consumption prediction. From the desirable experimental results,

we are full of confidence that taking the advantages of deep learning to effectively improve the energy efficiency of cloud DCs is a promising direction. Meanwhile, we will make endeavour in advanced architecture, intelligent data acquisition system, efficient computing framework, modelling and relevant algorithms.

8 Acknowledgments

This research work is supported by National Natural Science Foundation of China (grant nos. 61572525, 61602525) and Research Foundation of Education Bureau of Hunan Province, China (grant no. 19C1391). Also, the authors greatly appreciate the reviewers' valuable comments on this paper.

9 References

[1] Shuja, J., Balal, K., Madani, S.A., *et al.*: 'Survey of techniques and architectures for designing energy-efficient data centers', *IEEE Syst. J.*, 2016, **10**, (2), pp. 507–519

[2] Dayarathna, M., Wen, Y., Fan, R.: 'Data center energy consumption modeling: a survey', *IEEE Commun. Surv. Tutor.*, 2016, **18**, (1), pp. 732–794

[3] '2014 data center industry survey'. Available at https://journal.uptimeinstitute.com/2014-data-center-industry-survey/, accessed 14 July 2019

[4] Hasan, S., Alvares, F., Ledoux, T., *et al.*: 'Investigating energy consumption and performance trade-off for interactive cloud application', *IEEE Trans. Sustain. Comput.*, 2017, **2**, (2), pp. 113–126

[5] Xu, X.L., Dou, W.C., Zhang, X.Y., *et al.*: 'Enreal: an energy-aware resource allocation method for scientific workflow executions in cloud environment', *IEEE Trans. Cloud Comput.*, 2016, **4**, (2), pp. 166–179

[6] Jalali, F., Hinton, K., Ayre, R., *et al.*: 'Fog computing may help to save energy in cloud computing', *IEEE J. Sel. Areas Commun.*, 2016, **34**, (5), pp. 1728–1739

[7] Djemame, K., Bosch, R., Kavanagh, R., *et al.*: 'Paas-Iaas inter-layer adaptation in an energy-aware cloud environment', *IEEE Trans. Sustain. Comput.*, 2017, **2**, (2), pp. 127–139

[8] Chen, Y.W., Chang, J.M.: 'EMaas: cloud-based energy management service for distributed renewable energy integration', *IEEE Trans. Smart Grid*, 2015, **6**, (6), pp. 2816–2824

[9] Demirci, M.: 'A survey of machine learning applications for energy-efficient resource management in cloud computing environments'. Proc. 2015 IEEE 14th Int. Conf. on Machine Learning and Applications, Miami, FL, USA, December 2015, pp. 1185–1190

[10] Anastasopoulos, M., Tzanakaki, A., Simeonidou, D.: 'Stochastic energy efficient cloud service provisioning deploying renewable energy sources', *IEEE J. Sel. Areas Commun.*, 2016, **34**, (12), pp. 3927–3940

[11] Kurdi, H.A., Alismail, S.M., Hassan, M.M.: 'LACE: a locust-inspired scheduling algorithm to reduce energy consumption in cloud datacenters', *IEEE Access*, 2018, **6**, pp. 35435–35448

[12] Zhang, Y.C., Yang, R., Zhang, K.Q., *et al.*: 'Consumption behavior analytics-aided energy forecasting and dispatch', *IEEE Intell. Syst.*, 2017, **32**, (4), pp. 59–63

[13] Roy, S., Rudra, A., Verma, A.: 'An energy complexity model for algorithms'. Proc. 4th Conf. ITCS, Cambridge, MA, USA, 2013, pp. 283–304

[14] Tudor, B.M., Teo, Y.M.: 'On understanding the energy consumption of arm-based multicore servers'. ACM Sigmetrics/int. Conf. on Measurement and Modeling of Computer Systems, Pittsburgh, PA, USA, 2013, pp. 267–278

[15] Song, S.L., Barker, K., Kerbyson, D.: 'Unified performance and power modeling of scientific workloads'. Proc. 1st Int. Workshop E2SC, Denver, CO, USA, 2013, pp. 4:1–4:8

[16] Alan, I., Arslan, E., Kosar, T.: 'Energy-aware data transfer tuning'. Proc. 14th IEEE/ACM Int. Symp. CCGrid, Chicago, IL, USA, May 2014, pp. 626–634

[17] Tian, Y., Lin, C., Li, K.Q.: 'Managing performance andpower consumption tradeoff for multiple heterogeneous servers in cloud computing', *Cluster Comput.*, 2014, **17**, (3), pp. 943–955

[18] Mills, B., Znati, T., Melhem, R.: 'Energy consumption of resilience mechanisms in large scale systems'. Proc. 22nd Euromicro Int. Conf. PDP, Feb 2014, pp. 528–535

[19] Qiu, X.W., Dai, Y.S., Xiang, Y.P.: 'A hierarchical correlation model for evaluating reliability, performance, and power consumption of a cloud service', *IEEE Trans. Syst., Man Cybern.-Syst.*, 2016, **46**, (3), pp. 401–412

[20] Hameed, A., Khoshkbarforoushha, A., Ranjan, R., *et al.*: 'A survey and taxonomy on energy effcient resource allocation techniques for cloud computing systems', *Computing*, 2016, **98**, (7), pp. 751–774

[21] Li, Z., Tesfatsion, S., Bastani, S., *et al.*: 'A survey on modeling energy consumption of cloud applications: deconstruction, state of the art, and trade-off debates', *IEEE Trans. Sustain. Comput.*, 2017, **2**, (3), pp. 255–274

[22] Zhou, Z., Abawajy, J., Chowdhury, M., *et al.*: 'Minimizing SLA violation and power consumption in cloud data centers using adaptive energy-aware algorithms', *Future Gener. Comput. Syst.*, 2018, **86**, (8), pp. 836–850

[23] Khosravi, A., Andrew, A., Buyya, R.: 'Dynamic VM placement method for minimizing energy and carbon cost in geographically distributed cloud data centers', *IEEE Trans. Sustain. Comput.*, 2017, **2**, (2), pp. 183–196

[24] Zhao, H.S., Wang, Y.L., Zhan, Y., *et al.*: 'Practical model for energy consumption analysis of beam pumping motor systems and its energy-saving applications', *IEEE Trans. Ind. Appl.*, 2018, **54**, (2), pp. 1006–1016

[25] Mobius, C., Dargie, W., Schill, A.: 'Power consumption estimation models for processors, virtual machines, and servers', *IEEE Trans. Parallel Distrib. Syst.*, 2014, **25**, (6), pp. 1600–1614

[26] Zhou, Z., Abawajy, J.H., Li, F.M., *et al.*: 'Fine-grained energy consumption model of servers based on task characteristics in cloud data center', *IEEE Access*, 2018, **6**, pp. 27080–27090

[27] Zhang, X., Lu, J.J., Qin, X., *et al.*: 'A high-level energy consumption model for heterogeneous data centers', *Simul. Modelling Pract. Theory*, 2013, **39**, (8), pp. 41–55

[28] Bertran, R., Becerra, Y., Carrera, D., *et al.*: 'Energy accounting for shared virtualized environments under DVFS using PMC-based power models', *Future Gener. Comput. Syst.*, 2012, **28**, (2), pp. 457–468

[29] Park, S., Mun, Y.: 'Prediction method about power consumption by using utilization rate of resource in cloud computing environment'. Int. Conf. on Big Data & Smart Computing, Hong Kong, China, 2016, pp. 265–268

[30] Liu, H., Yan, F., Zhang, S., *et al.*: 'Source-level energy consumption estimation for cloud computing tasks', *IEEE Access*, 2018, **6**, pp. 1321–1330

[31] Yu, J.Y., Hu, Z.G., Zhou Z., : 'A CMP energy consumption estimate model for computer systems', *J. Univ. Electron. Sci. Technol. China*, 2015, **44**, (3), pp. 422–427

[32] Foo, Y.W., Goh, C., Li, Y.: 'Machine learning with sensitivity analysis to determine key factors contributing to energy consumption in cloud data centers'. Int. Conf. on Cloud Computing Research and Innovations (ICCCRI), Singapore, Singapore, 2016, pp. 107–113

[33] Foo, Y.W., Goh, C., Lim, H.C., *et al.*: 'Evolutionary neural network based energy consumption forecast for cloud computing'. Int. Conf. on Cloud Computing Research and Innovation, Singapore, Singapore, 2015, pp. 53–64

[34] Foo, Y.W., Goh, C., Lim, H.C., *et al.*: 'Evolutionary neural network modeling for energy prediction of cloud data centers'. Int. Symp. on Grids and Clouds (ISGC), Taipei, Taiwan, China, 2015

[35] Kawaguchi, S., Yachi, T.: 'Adaptive power efficiency control by computer power consumption prediction using performance counters', *IEEE Trans. Ind. Appl.*, 2016, **52**, (1), pp. 407–413

[36] Salcedo-Sanz, S., Cornejo-Bueno, L., Prieto, L., *et al.*: 'Feature selection in machine learning prediction systems for renewable energy applications', *Renew. Sust. Energy Rev.*, 2018, **90**, pp. 728–741

[37] Salem, N., Hussein, S.: 'Data dimensional reduction and principal components analysis', *Procedia Comput. Sci.*, 2019, **163**, pp. 292–299

[38] Laohakiat, S., Phimoltares, S., Lursinsap, C.: 'A clustering algorithm for stream data with LDA-based unsupervised localized dimension reduction', *Inf. Sci.*, 2017, **381**, pp. 104–123

[39] Sun, B.Y., Zhang, X.M., Li, J., *et al.*: 'Feature fusion using locally linear embedding for classification', *IEEE Trans. Neural Netw.*, 2009, **21**, (1), pp. 163–168

[40] 'Entropy (information theory)'. Available at https://en.wikipedia.org/wiki/Entropy_(information_theory), accessed 14 July 2019

[41] LeCun, Y., Bengio, Y., Hinton, G.: 'Deep learning', *Nature*, 2015, **521**, (7553), pp. 436–444

[42] Ahmad, F., Chakradhar, S.T., Raghunathan, A., *et al.*: 'Tarazu: optimizing mapreduce on heterogeneous clusters'. Int. Conf. on Architecture Support for Programming Languages and Operating System, London, UK, 2012, vol. 40, no. 1, pp. 61–74