Machine Learning with Sensitivity Analysis to Determine Key Factors Contributing to Energy Consumption in Cloud Data Centers

Yong Wee Foo[1,2], Cindy Goh[1], Yun Li[1]
[1]School of Engineering, University of Glasgow, Glasgow, U.K.
[2]School of Engineering, Nanyang Polytechnic, Singapore
[2]Foo_Yong_Wee@nyp.edu.sg

***Abstract*—Machine learning (ML) approach to modeling and predicting real-world dynamic system behaviours has received widespread research interest. While ML capability in approximating any nonlinear or complex system is promising, it is often a black-box approach, which lacks the physical meanings of the actual system structure and its parameters, as well as their impacts on the system. This paper establishes a model to provide explanation on how system parameters affect its output(s), as such knowledge would lead to potential useful, interesting and novel information. The paper builds on our previous work in ML, and also combines an evolutionary artificial neural networks with sensitivity analysis to extract and validate key factors affecting the cloud data center energy performance. This provides an opportunity for software analysts to design and develop energy-aware applications and for Hadoop administrator to optimize the Hadoop infrastructure by having Big Data partitioned in bigger chunks and shortening the time to complete MapReduce jobs.**

Keywords—machine learning, artificial neural networks, sensitivity analysis, cloud computing, energy efficiency, genetic algorithm

I. Introduction

The accelerated growth in cloud computing is expected to drive energy consumption of cloud data centers to new highs. An effective engineering ratio that measures the data center energy efficiency is the Power Usage Effectiveness (PUE). This term records baseline data and traces energy efficiency movements. It is expressed as a ratio as shown in Eq. 1, with the overall energy efficiency improving as the value decreases towards 1.

$$PUE = \frac{\sum(P_{mech}+P_{elect}+P_{compute}+P_{others})}{\sum P_{compute}} \qquad (1)$$

where, the numerator denotes the sum of all power consume by the cloud data center including the mechanical facility (chillers and computer room air-con or CRAC), the electrical facility (switchgear, UPS, battery backup), the ICT computing infrastructure (servers, storage, networks and telecommunications equipment) plus any other devices (lightings, printers, personal computers, VoIP phones, fax machines and etc.) expend to support the cloud data center operations, and the denominator denotes the sum of all power consume by the ICT computing infrastructure only that produces useful IT work.

Based on Uptime Institute's 2014 Data Center Industry Survey, the average data centers' PUE has only improved slightly from 1.89 in 2011 to 1.7 in 2014 [1]. This is a gain of 11.2% compared to a gain 32.3% from PUE of 2.50 to 1.89 from 2007 to 2011. The survey also reported that 77% of the participating industry cited that the management has set a target PUE, with more than half expecting to lower PUE to 1.5 or better. This scenario presents a tremendous opportunity for researchers, engineers and technologists to further improve the cloud data center energy efficiency.

Recently, applying data-driven ML techniques to lower cloud data center energy consumption have been a hot research topic. Gao [2], proposed improving the cloud data center Power Usage Effectiveness (PUE) using ML technique based on artificial neural networks (NN). The feed-forward NN takes in 19 inputs variables, contains 5 hidden layers with 50 nodes per layer and outputs 1 variable. The NN model has achieved a high predictive accuracy with a mean absolute error of 0.004 and standard deviation of 0.005 on the test dataset. The model was validated with a 'live' experiment conducte d by simulating an actual increase in process water supply temperature to the server floor by 3^{o}F (or ~1.7^{o}C) resulting in an expected decrease of ~0.005 PUE as predicted by the model.

Chen [3] suggested a spatially-aware Virtual Machine (VM) workload placement method, called SpAWM to optimize the consumption of power and cooling in cloud data center. SpAWM adopts the ML approach using neural network and reinforcement learning (RL). Developed from Markov Decision Process (MDP), the central idea in RL is to learn the optimal *action*, a_t via a trial and error process, after taking into account every *state*, s_t visited by the system. For every state-action pair, the RL keeps an associated Q value. If the selected action in a state is positive, the feedback increases the Q value. Else, if the feedback is negative, Q value is decreased. In the RL, the new Q value is updated via the Eq. 2,

$$Q(s_t, a_t) = \alpha\{r_t + \lambda[Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]\} \qquad (2)$$

where, $Q(s_{t+1}, a_{t+1})$ denotes the Q value of the next state *t+1*, $Q(s_t, a_t)$ denotes the Q value of the current state *t*, α is the learning rate, λ is the discount factor and r_t is the immediate reward received at state *t*. The number of Q values can grow exponentially if the state-action pair is large, hence NN modeling is utilized for the cloud data center environment to capture the relationship between resource utilization (state space), workload assignments (actions) and thermal distribution (reward function). In the paper, the NN RL is designed to optimize the objective function reward R, which is expressed in Eq. 3,

$$R = T_{th}^{in} - max\{T_i^{in}\}, i \in [1, n] \qquad (3)$$

where, T_{th}^{in} is the safe threshold for server inlet temperature and $\max\{T_i^{in}\}$ is the maximum observed server inlet temperature for the i^{th} server, their difference being resulted in R or the adjustable temperature for the CRAC. Therefore, the higher the R, the higher the savings for cooling energy. During the experiment, the environment is constantly monitored while VM workloads are being distributed in an energy-efficient manner by SpAWM to maintain a high R. The ML approach employs a backpropagation feed-forward NN with 6 input variables, 1 hidden layer of 20 nodes, and 6 outputs. The inputs variables are the server resource utilization states and the outputs are inlet temperature, to be predicted. The experiment is based on the data collected from 6 blade server enclosures from each rack up to a total of 10 racks. Result from the simulated workload showed that NN RL is able to accurately predict the inlet server temperature, enabling SpAWM with energy-aware capability to optimize VM placement to servers. The trial and error nature of RL suggests there would be initial penalty in the form of wasted energy due to the 'exploratory' nature of the algorithm. That is, VMs workloads may be assigned to less optimal servers in order to 'explore' potential solution space in search for better 'rewards'. Premature convergence or convergence to a local minimum (sub-optimal convergence) could also dampen energy savings as tuning the learning rate and discount factor to avoid such, is an expensive process.

Tarutani et al. [4] applied ML technique using regression models to predicting the cloud data center temperature distribution. The power consumption is then reduced via pro-active control of the server load and tuning the CRAC settings. The inputs to the model consist of power consumption of servers, server intake air temperature, UPS power consumption, current temperature distribution and its temporal changes, CRAC outlet air temperature, and CRAC air volume. The inputs are denoted by $x(t) = (x_1(t), x_2(t), \ldots, x_N(t))$ where N is the number of sensor inputs or cloud data center operation parameters at time-step *t*. The output is the predicted temperature distribution in the cloud data center at time-step *t*, as denoted by $y(t) = (y_1(t), y_2(t), \ldots, y_M(t))$ where M is the number of temperature sensors. The predicted temperature, $\hat{y}_k(t + \Delta t)$ at time-step $t + \Delta t$, is given by Eq. 4,

$$\hat{y}_k(t + \Delta t) = y_k(t) + F_k(x(t)) \qquad (4)$$

where, $y_k(t)$ is the temperature obtained at time-step *t* by the k^{th} sensors and $F_k(x(t)$ is function for predicting the temperature of the k^{th} sensor given the input variables at time-step *t*. The sum of squared error is given by Eq. 5,

$$e_k = \sum_1^n (\hat{y}_k(t + \Delta t) - y_k(t + \Delta t))^2 \qquad (5)$$

where, *n* is the number of training dataset. However, the large number of input variables has compelled a reduction in the data dimensionality. Working with the transformed data whereby the input dataset is significantly reduced improves the learning process. A data compression technique using Principle Component Analysis (PCA) is then applied in addition to the regression model. Since there are correlations among input variables, PCA basically compress the data by expressing the data in terms of the patterns between the inputs. The components of *x(t)* are reduced to values denoted by $p(t)=(p_1(t), p_2(t), \ldots, p_C(t))$ *where* $C \ll N$, is the number of feature values. With this, Eq. (4) can be rewritten as:

$$\hat{y}_k(t + \Delta t) = y_k(t) + F_k{}'(p(t)) \qquad (6)$$

Finally, Tarutani et al. compared the prediction model with random forest method that utilizes decision-tree as a weak learner to avoid overfitting and to increase the accuracy and speed of the prediction. From the result, it shows that the number of input features selected affects the predictive accuracy of both the linear regression method and the random forest method. Higher number of feature inputs leads to lesser predictive accuracy. As the number of sensors in the data center grows, which is inevitable, feature selection to reduce the number of inputs would become a challenge.

In this paper, we apply the ML approach combining evolutionary NN and SA to model the energy consumption of a dynamic cloud data center. Our approach employs a Genetic Algorithm (GA) for feature selection utilizing SA as a guide. The feature subset, along with the NN architecture, is represented by a structurally-inclusive encoding scheme in the form of a chromosome matrix. A population of chromosomes is maintained through the genetic process of crossover and mutation. The chromosome's fitness is evaluated at every generation to determine its survival in the next generation. The algorithm "prunes" away connections between the neurons to deemphasize a particular input neuron's contribution to the NN's output should such an input feature causes the chromosome to have a weak fitness. The eventual NN is a network with reduced complexity. This leads to a model with better generalization and at lower computational cost. A complex NN has high computational cost and the tendency to overfit. The proposed evolutionary NN combined with SA helps to extract the key factors impacting the cloud data center energy performance. This information provides insights for better decision-making and management of the cloud data center energy consumption. The rest of the paper is organized as follows: Section II explains the evolutionary NN, Section III describes the data collection and SA approach, the experiments and results are discussed in Section IV and Section V concludes with recommendations for future work.

II. EVOLUTIONARY NEURAL NETWORK

Machine learning approach to reducing energy consumption in cloud data center is a viable solution. To appreciate the interest in this hot topic relating to ML applications for energy efficient management in cloud computing environment, one may refer to the survey papers by Demirci [5], Tantar [6] and Zhan [9]. Evolutionary NN as a ML approach to modeling and predicting non-linear dynamic system such as the cloud data center is a powerful and promising approach. However, one area that has not been adequately address is the area of establishing impacts of the inputs to a FFNN, in particular, our interest is to discover which input features have the most impact to the cloud data center energy performance. Our approach is to apply evolutionary algorithm to detect the 'weightier' input features that contribute positively to the NN's fitness, directed by SA. As NNs are black-box models, the technique itself prevents any easy analysis of the relationships between the inputs and outputs. In NN modeling, every computational iteration ends up with new and different connection weights. The results in these different weight settings can be nearly or totally the same. This is because each starting weight matrices are different and during the training, the number of free degree is very high. The numerical input-output relationships can be satisfied by separate groups of neurons, weights and connections. This is the short-coming for a black-box modeling approach. Hence, establishing impacts of the inputs to a FFNN is non-trivial.

A. *Multi-Layer Feed-Forward Neural Network*

The NN architecture used for modeling the cloud data center is a multi-layer feed-forward neural network (FFNN). It has three layers; namely the input layer, the hidden layer and the output layer. The input layer has a total of 12 input nodes and 1 bias node. The input nodes represent 12 energy-related variables of the Hadoop cluster. The hidden layer has a maximum of 20 hidden nodes and 1 bias node. The output layer has 1 output node. Table 2 summarizes the description of the NN inputs and output and Fig. 1 depicts the NN architecture. During the evolution process, the number of NN connections, the connection weights and the number of hidden layer neurons, are determined. GA explores the solution space in search of the fittest or the optimal NN structure.

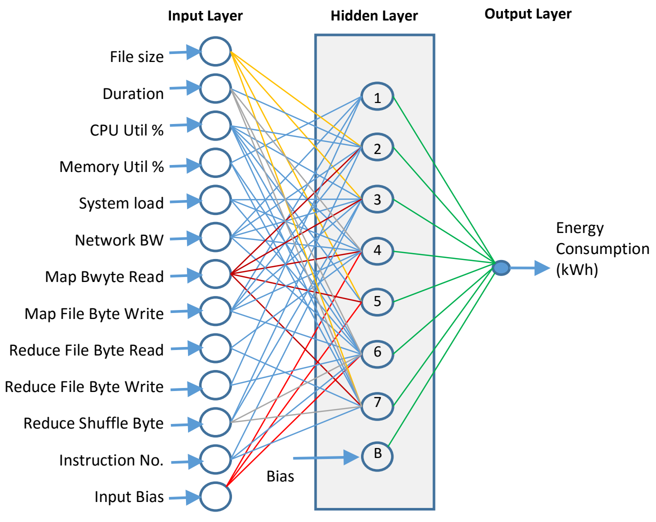

Fig. 1. NN Architecture with Input Features

B. *Genetic Algorithm for NN Optimization*

Table 1 depicts a chromosome matrix example which is encoded to represent the problem in the GA solution space. The chromosome matrix is an individual, known also as the genotype, which has a corresponding mapping to its phenotype as shown in Fig. 2. A population of these individuals is maintained with each chromosomes representing a possible solution in the GA search space. The optimal solution will be the fittest individual over many cycles of genetic evolution. The optimum NN structure is the phenotype mapping of the fittest genotype.

TABLE 1 A CHROMOSOME MATRIX EXAMPLE

	Hidden node 1	Hidden node 2	Hidden node 3	Hidden node 4
Input node 1	0	-0.348	0	0.492
Input node 2	0	0	0.492	0.214
Input node 3	0.628	0	0.914	0
Bias node	-0.583	-0.569	0.239	-0.921
Output node	0.023	-0.345	0.295	0.148

Embodied in the chromosome matrix in Table 1 are the weights and connection characteristics. In this example, the input weights (denoted by values in the first 3 rows) represent the corresponding links between the input nodes and the hidden nodes. The output weights (denoted by values in the last row) represent the corresponding links between the output node and the hidden nodes. A 'zero' value represents no connectivity between the corresponding nodes. For instance, the input weights in the matrix positions (1,1), (1,3), (2,1), (2,2), (3,2) and (3,4) are zeroes, it means that there are no connection between the corresponding input nodes and the hidden nodes. And if the one of the output weight is zero, this connection, or column, can be ignored as it will not affect the output in any way. If all the input for that hidden node is zero, regardless of the output weight, that connection will also be ignored as there is no neuron activation of that hidden node to the output node. Another matrix with the redundant columns removed will be stored to reduce computation from recalculating the matrix. The matrix with the redundant columns is still kept as its dimension is required for crossover and any residual values might be important for the crossover.

The NN input layer comprises of n features denoted as $(X_1\, X_2, \ldots\, X_n)$ During the NN learning process, the evaluation of the intensity of the stimulation (excitatory or inhibitory) from the neurons of the preceding layer is expressed in Eq. 7,

$$a_j = \sum_{i=1}^{n} X_i W_{ij} \tag{7}$$

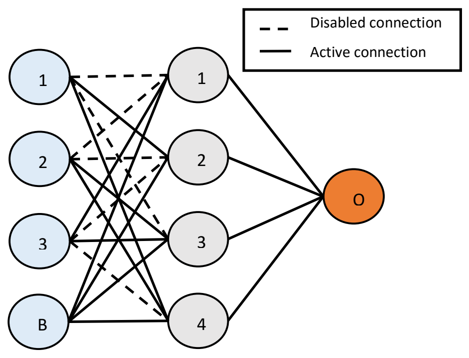

Fig. 2. NN Phenotype Mapping from its Corresponding Genotype

TABLE 2 NN INPUT FEATURE SUBSET PRESENTING THE HADOOP CLUSTER

Category		Metric	Unit	Description	Method of collection
Input	System	1. CPU utilization	%	% CPU time on MapReduce process	Ganglia
		2. System Load	%	% system load on MapReduce process	Ganglia
		3. Memory use	%	% memory use for MapReduce process	Ganglia
	IO	4. Map file byte read 5. Reduce file byte read 6. Map file byte write 7. Reduce file byte write	Gigabyte Gigabyte Gigabyte Gigabyte	Data read by Map from local disk Data read by Reduce from local disk Data written by Map to local disk Data written by Reduce to local disk	Hadoop built-in counters
	Network Transfer	8. Reduce Shuffle bytes	Gigabyte	Data transferred from Map to Reduce	Hadoop built-in counters
		9. Network Bandwidth	Gigabit per sec	Data transmitted and received	Ganglia
	Job Profile	10. No. of MapReduce Instructions	Number	Job's instruction number	Ganglia
		11. File size	Gigabyte	Size of MapReduce jobs	Hadoop built-in counters
		12. Job completion duration	Hour	Time taken to finish a MapReduce job	Hadoop built-in counters
Output	Energy	1. Energy consumption	kWh	Energy consumed by Hadoop cluster	SNMP on iPDU

where, a_j is the activation function of the j^{th} downstream neuron, X_i is the output value of the i^{th} neuron at the previous layer and W_{ij} is the connection weight between the i^{th} neuron of the previous layer and the j^{th} neurons of the current layer. The activation function implemented is the sigmoid function shown in Eq. 8.

$$f(a_j) = \frac{1}{1+e^{-a_j}} \tag{8}$$

The objective function to minimize (or to maximize in the case of the fitness function) during the NN training is the mean squared error (MSE) given by Eq. 9,

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(Y - \hat{Y})^2 \tag{9}$$

where, Y is the target at the output of the NN and $\hat{Y}$ is the actual calculated value by the NN and m is the number of samples. This fitness indicates how good the chromosome is in comparison with the other solutions in the population. The chromosomes compete for survival. Thus, the higher the fitness value, the higher the chances of survival, reproduction and representation in the subsequent generation.

The GA optimization of NN starts with an initial set of random potential solutions, expressed as a population of chromosomes. In order to create the subsequent generation, chromosomes from the previous generation are merged using the crossover operation or modified by using the mutation operator. These processes populate the subsequent generation with new chromosomes, also known as offspring. Fitter chromosomes are selected and weaker chromosomes are rejected to keep the population size constant and to maintain the overall health of the population on a progressing level. After repeating the process for several generations, the best chromosome will emerge, representing the optimum or suboptimal solution to the problem.

III. DATA COLLECTION AND SENSITIVITY ANALYSES

A. *Experiment Setup and Data Collection*

A Hadoop cluster with Hadoop Distributed File System (HDFS) and MapReduce stack (Facebook Apache Hadoop version 0.20.1), is set up to perform the experiments. The software stack is installed over 6 x HP Proliant DL360P and DL380P Gen8 servers, consisting of 120 cores housed within a single rack. Each server is equipped with 64 GB memory, dual socket 6-core Intel(R) Xeon(R) CPU E5-2667 @ 2.90GHz with hyper-threading technology. The Hadoop cluster comprises of 1 x namenode, 1 x secondary namenode and 4 x datanodes. All nodes are installed with CentOS 6.5 and running on bare-metal hardware without hypervisor or virtualization. A top-of-rack (TOR) Gigabit Ethernet switch connects the nodes at 1Gigabit per second speed. A mixture of MapReduce jobs, in the form of the WordCount application and Sort application are executed during the experiments. The Hadoop MapReduce counters such as the Map file byte read, the Reduce file byte write and etc. are extracted using the build-in Hadoop web admin user interfaces (UIs). The counters can be access via the HDFS namenode admin at port 50070 and the MapReduce Job tracker admin at port 50030. The other counters such as the CPU and memory utilization and network IO are collection using Ganglia, an open source monitoring system. The power consumption data is collected using the Raritan intelligent Power Distribution Unit (iPDU), through which the servers' power supply are connected into. The data collected from the Hadoop cluster are used to train and calibrate the NN models. The details of the testbed setup and evolutionary NN training is described in our earlier work in [7][8].

B. *Sensitivity Analysis*

Various authors have explored various SA techniques in determining which inputs in NN are significant [11][12][13][14][15]. In [10], a series of seven different SA methods were reviewed. Amongst these methods, two are of particular interest. They are; the Partial Derivatives (PaD) method which consists of calculating the partial derivatives of the output according to the input variables, and the 'weights' method which is a technique for partitioning the connection weights to determine the relative importance of the various inputs.

1) Partial Derivatives Method

The PaD method [16][17], allows contribution analysis of the inputs by providing a profile of the variation of the output for small changes of one input variable. For instance, in a network with n_i inputs, one hidden layer with n_h neurons, one output n_o=1 and using the logistic-sigmoid activation function, the partial derivatives of the output y_i with respect to input x_j (with j=1, …, N and N the total number of samples) is given by the following relationship in Eq. 10,

$$d_{ji} = S_j \sum_{h=1}^{n_h} w_{ho} I_{hj} \left(1 - I_{hj}\right) w_{ih} \qquad (10)$$

where, S_j is the derivative of the output neuron with respect to its input, I_{hj} is the response of the h^{th} hidden neuron, w_{ho} is the weight between the h^{th} hidden neuron and the output neuron and w_{ih} is weight between the i^{th} input neuron and the h^{th} hidden neuron. If the partial derivative is negative, which indicates negative impact, that is, as the input variable increases, the output variable will decrease. On the contrary, if the partial derivative is positive, which indicates a positive impact, that is, as the input variable increases, the output variable will increase. The relative contribution of an input to the NN output can be obtained via the sum of square derivatives (SSD) value as expressed in Eq. 11. The input variable that has the highest SSD value, has the highest impact on the output variable.

$$SSD_i = \sum_{j=1}^{N} \left(d_{ji}\right)^2 \qquad (11)$$

2) 'Weights' Method

The 'weights' method [18], [19] is a technique for segregating the connection weights to accord relative importance to the inputs. The connection weights between the hidden node and the output node are associated with the connection weights of the input nodes to that hidden node. For example, in a NN with n input nodes, m hidden nodes and 1 output node, the 'weights' method is implemented using the following relationships in Eq. 12,

$$Q_{ih} = \frac{|w_{ih} . w_{ho}|}{\sum_i |w_{ih} . w_{ho}|} \qquad \textit{(for i=1,..,n; h=1,..,m)} \qquad (12)$$

where, w_{ih} is the connection weight between the input node i and the hidden node h, w_{ho} is the connection weight between the hidden node h and the output node o, and Q_{ih} is the weighted influence of individual hidden node h to the output given its association with all its inputs i, and in Eq. 13,

$$RI_i(\%) = \frac{\sum_h Q_{ih}}{\sum_h \sum_i Q_{ih}} \text{x}100 \qquad \textit{(for i=1,..,n; h=1,..,m)} \qquad (13)$$

where, RI_i is the relative importance of input node i in percentage term.

IV. Experiments and Results

Though both the PaD method and the 'weight' method are able to provide ML with explanatory capability [10], the 'weight' method is adopted in our implementation as it combines more readily with our GA chromosome matrix encoding. Table 3 illustrates, by means of the chromosome matrix described in Table 1, our GA with SA approach to determine the relative importance of the input features on the output. By applying Eq. 11 and Eq. 12, it can be seen that input node 1 has the highest relative importance at 42.4%, followed by input node 3 at 41.3% and input node 2 at 16.3%.

Table 3 Example Chromosome Matrix for SA using 'Weights' Method

	Hidden node 1	Hidden node 2	Hidden node 3	Hidden node 4	Sum impact of input node at output or RI(%)
Input node 1	Q_{11}=0	Q_{21}=1.0	Q_{31}=0	Q_{41}=0.697	1.697 or 42.4%
Input node 2	Q_{12}=0	Q_{22}=0	Q_{32}=0.350	Q_{42}=0.303	0.653 or 16.3%
Input node 3	Q_{13}=1.0	Q_{23}=0	Q_{33}=0.650	Q_{43}=0	1.650 or 41.3%

A. Actual Data

Integrating the 'weights' method into our evolutionary NN approach, the RI of the input variables from the Hadoop dataset is plotted for further analysis. In the experiment, we set the GA population size to 100 chromosomes for 100 generations. At each generation, the chromosomes fight for survival by evolving towards an optimal solution through the process of selection, crossover and mutation. The fittest chromosome from each generation is selected to compute the inputs' RI. The data is collected and averaged over 100 generations. The process is repeated for 20 runs and the result is shown in Fig. 3 and Table 5.

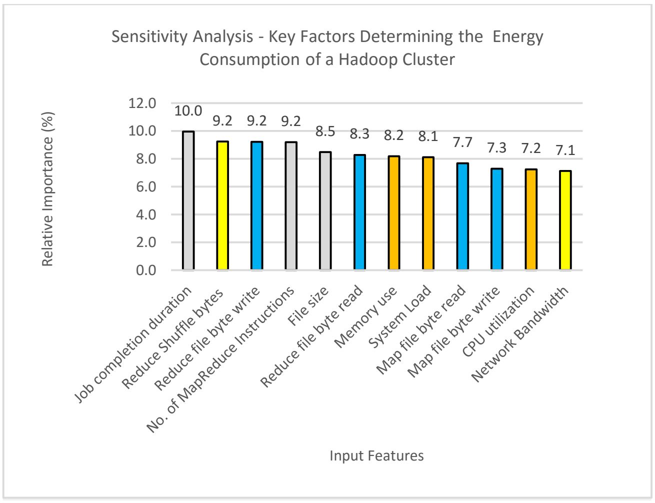

Fig. 3. Relative Importance of Input Variable on the Energy Consumption

From Fig. 3, it is observed that 'Job duration' ranks the highest with RI at 10.0%. The next three variables of importance are 'Reduce shuffle bytes', 'Reduce file byte write' and 'Number of mapreduce instructions', with RI at 9.2% each. The 'Reduce shuffle bytes' counter keeps track of the total bytes being shuffled which is an indicator of high presence of network-intensive activities. At the shuffle stage of the MapReduce process, files are shuffled from mappers to reducers. Depending on the nature of the task, as in the case of the Sort tasks, shuffle activities could be intensive. The variables 'Memory use', 'System load' and 'CPU utilization' are ranked 7th, 8th and 11th respectively. This may seem as a surprise at first, which shall be explained with Fig. 4 in the later section. Another observation in Fig. 3 is that the input variable 'file size' is ranked 5th. As MapReduce is a distributed and parallel processing framework where the Big Data files are split into chunks of HDFS blocks. The larger the files the more chunks to be distributed into HDFS blocks residing at various datanodes. The mapper and reducer daemons in the datanodes perform the MapReduce tasks on the data as assigned by the Job tracker. The input variable 'file size', in this case, has a fairly high RI as it is a key factor contributing to the energy consumption. The variable 'Network bandwidth' is the least in relative importance. This counter keeps track of the total number of bytes sent and received in the Hadoop cluster.

In Fig. 4, the input variables are grouped by categories with the breakdown shown in Table 4. It is observed that the 'I/O' category has the most impact over energy consumption, followed by 'Job profile'. The 'System' category, which includes utilization of CPU and memory, and OS processes, is ranked third. The least in relative importance is the 'Network transfer' category. Earlier in Fig. 3, it was noted that the input variables 'Memory use', 'System load' and 'CPU utilization' are ranked 7th, 8th and 11th respectively. However in the category chart in Fig. 4, 'System' taken as a whole, is ranked closely behind 'Job profile'.

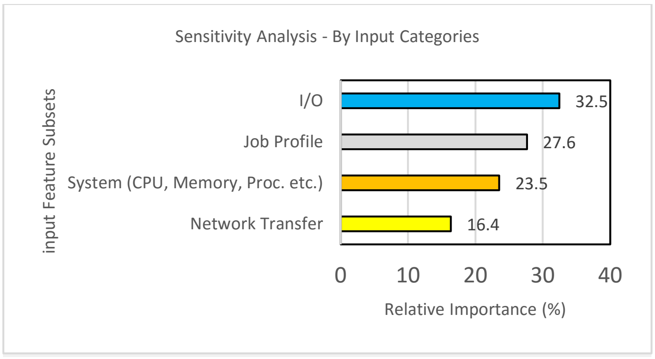

Fig. 4. Relative Importance of Input Features Grouped by Categories

Fig. 5 shows the RI plots for each of the twelve input variables of the NN. The RI is calculated from the fittest chromosome structure of that generation and traced over 100 generation or until convergence is reached, i.e. the optimal solution is found.

TABLE 4: INPUT VARIABLES GROUPED BY CATEGORIES

Category	Input Variables
System	CPU Utilization
	Memory Use
	System Load
I/O	Map file byte Read
	Map file byte Write
	Reduce file byte Read
	Reduce file byte Write
Network Transfer	Network Bandwidth
	Reduce Shuffle Bytes
Job Profile	Job completion duration
	File size
	Number of MapReduce Instructions

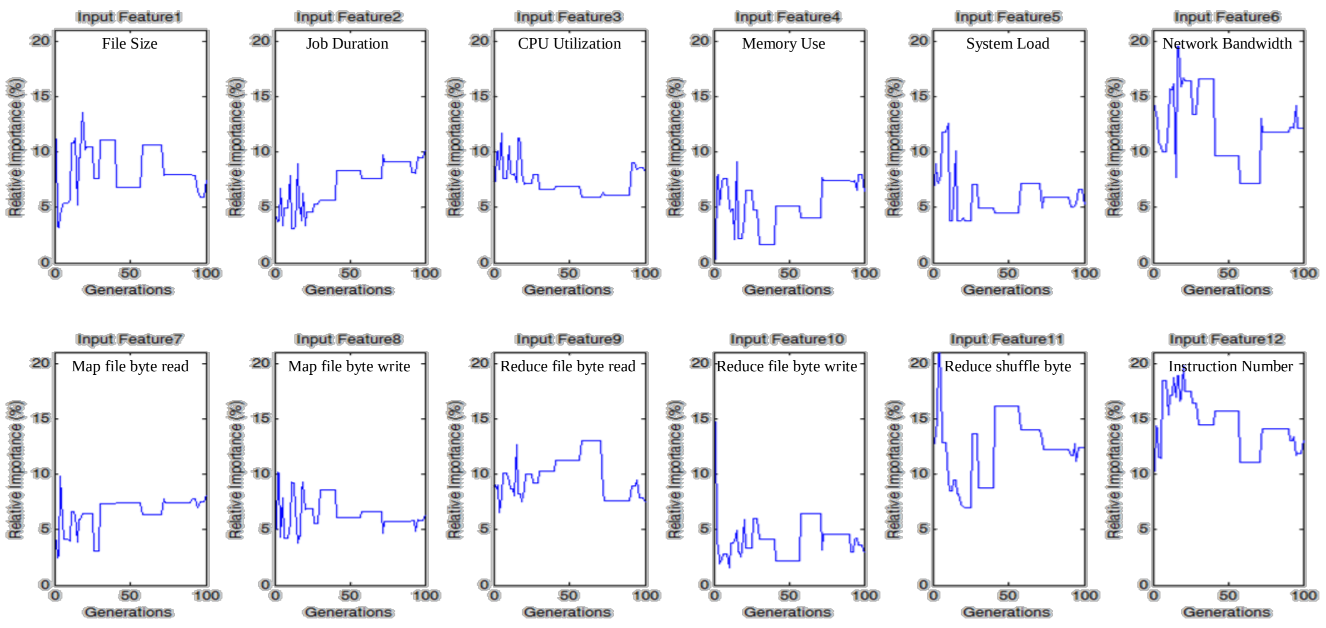

Fig. 5. Relative Importance of Inputs in Sensitivity Analysis of Neural Networks using 'Weighted' Method Combined with Genetic Algorithm

TABLE 5: COMPUTATION OF RI (%) FOR 20 RUNS

Run	Input Variable 1	Input Variable 2	Input Variable 3	Input Variable 4	Input Variable 5	Input Variable 6	Input Variable 7	Input Variable 8	Input Variable 9	Input Variable 10	Input Variable 11	Input Variable 12
1	7.6	16.4	4.4	7.5	13.9	4.0	6.1	9.1	5.9	10.2	8.4	6.4
2	10.7	9.5	5.8	12.0	6.8	7.0	8.4	9.3	5.8	9.1	7.2	8.5
3	7.9	4.5	7.7	11.3	7.6	7.0	10.3	7.4	6.2	7.9	8.8	13.5
4	9.9	5.3	12.3	7.9	7.6	5.8	8.3	9.0	8.3	6.6	6.9	12.2
5	8.5	7.1	7.2	5.3	5.9	12.1	6.6	6.5	9.8	4.1	12.4	14.5
6	10.8	7.2	8.3	7.7	6.7	6.3	7.1	6.3	9.2	13.5	8.5	8.5
7	8.0	8.0	7.1	10.5	7.5	6.9	11.0	12.1	7.3	9.5	5.5	6.5
8	11.4	9.6	6.2	7.2	10.5	9.7	7.6	8.8	6.8	6.1	8.6	7.6
9	7.3	9.2	6.5	8.4	9.7	6.3	10.2	5.8	10.1	6.3	12.4	7.7
10	11.4	8.5	4.1	7.1	7.9	2.6	6.8	10.1	6.5	15.5	12.3	7.3
11	7.0	12.0	7.6	3.1	8.9	12.8	6.4	6.4	10.9	8.5	7.5	8.9
12	7.9	8.3	7.9	7.2	7.9	8.1	8.4	5.6	6.7	11.0	13.6	7.4
13	5.4	10.3	5.4	9.5	11.3	6.1	4.3	7.8	7.8	9.9	15.4	6.8
14	4.1	10.3	6.8	7.7	6.9	7.8	8.5	3.5	8.6	12.9	10.9	12.2
15	8.6	14.7	6.5	12.9	10.9	5.1	8.7	5.5	8.2	7.3	4.6	7.1
16	6.9	12.2	7.1	9.4	6.7	8.3	6.7	8.7	9.2	9.0	8.3	7.5
17	8.1	10.6	8.3	10.4	5.8	7.7	7.7	7.2	10.6	9.2	5.9	8.6
18	8.6	12.3	7.7	8.5	6.2	9.2	5.6	3.0	9.7	8.7	9.6	11.0
19	10.5	13.1	7.1	3.8	6.1	5.3	6.6	4.8	9.6	9.3	9.7	14.1
20	9.2	10.1	10.9	6.3	7.7	4.5	8.1	8.6	8.6	9.7	8.3	8.0
Ave	8.5	10.0	7.2	8.2	8.1	7.1	7.7	7.3	8.3	9.2	9.2	9.2
Stdev	1.8	1.7	1.7	1.6	1.5	1.8	1.6	1.6	1.7	1.7	2.0	1.7

V. CONCLUSION AND FUTURE WORK

The ability to give meaningful insights to black-box NN models can provide explanation on how the various system parameters affect its output. This benefit is of interest as useful and novel information can be derived to improve system characteristics and performance. In this paper, we have presented an approach combining evolutionary NN with sensitivity analysis. It was shown that the 'weights' method is able to seamlessly integrate with our GA to determine key factors contributing to energy consumption in cloud data centers. The results show that IO activities contribute most significantly to energy consumption. Armed with this insight, data centers can reduce energy consumption by minimizing access to storage or utilizing more efficient storage infrastructures and components. The other aspect revealed in our experiment is that the job profile such as file size, time taken to complete a task and the resources assigned to the job have relatively high impact on the energy consumption. As jobs may be bounded by service-level-agreement (SLA), adequate resources are usually assigned to complete the tasks within a certain time. Hence, this knowledge provides an opportunity for software analyst to design and develop energy-aware applications that optimizes resource allocation. Another potential energy savings that could be realized is to partition the Big Data into bigger chunks and place them closest to the mappers and reducers in the Hadoop HDFS. By doing so would shorten the time to complete the MapReduce jobs and thus saving energy.

Although our experiment combining GA and SA has managed to shed light on key factors contributing to energy consumption, further investigations are still needed. For future developments, we plan to take advantage of ML black-box approach with SA to develop a 'grey box' that will preserve the physical significance of the system parameters while at the same time model the nonlinear complexities of the cloud data center to allow for greater energy savings and better understanding of the system.

REFERENCES

[1] Uptime Institute, 2014 Data Center Industry Survey.

[2] J. Gao, "Machine learning applications for data center optimization," Research at Google, 2014.

[3] H. Chen, M. Kesavan, K. Schwan, A. Gavrilovska, P. Kumar and Y. Joshi, "Spatially-aware optimization of energy consumption in consolidated data center systems," Proceedings of the ASME 2011 Pacific Rim Technical Conference and Exhibition on Packaging and Integration of Electronic and Photonic Systems, 2011, pp. 461–470.

[4] Y. Tarutani, K. Hashimoto, G. Hasegawa, Y. Nakamura, T. Tamura, K. Matsuda and M. Matsuoka, "Temperature Distribution Prediction in Data Centers for Decreasing Power Consumption by Machine Learning," IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom), 2015, pp. 635-642.

[5] M. Dermirci, "A Survey of Machine Learning Applications for Energy-Efficient Resource Management in Cloud Computing Environments," IEEE 14th International Conference on Machine Learning and Applications, 2015, pp. 1185-1190.

[6] A.-A. Tantar, A. Q. Nguyen, P. Bouvry, B. Dorronsoro and E.-G. Talbi, "Computational Intelligence for Cloud Management Current Trends and Opportunities," in Proceedings of the IEEE Congress on Evolutionary Computation (CEC), 2013, pp. 1286-1293.

[7] Y. W. Foo, C. Goh, H. C. Lim, Z. H. Zhan and Y. Li, "Evolutionary Neural Network Based Energy Consumption Forecast for Cloud Computing," Proceedings of the 2015 International Conference on Cloud Computing Research and Innovation, 2015, pp. 53-64.

[8] Y.W. Foo, C. Goh, H.C. Lim and Y. Li, "Evolutionary Neural Network Modeling for Energy Prediction of Cloud Data Centers," International Symposium on Grids and Clouds, 2015.

[9] Z. H. Zhan, X. F. Liu, Y. J. Gong, J. Zhang, S.H. Chung, and Y. Li., "Cloud computing resource scheduling and a survey of its evolutionary approaches," ACM Computing Surveys, vol. 47, no. 4, Article 63, pp. 1-33, Jul. 2015.

[10] M. Gevrey, I. Dimopoulos and S. Lek, "Review and comparison of methods to study the contribution of variables in artificial neural network models," Ecological Modelling, 2003, vol. 160(3), pp. 249-264.

[11] P. Cortez and M. J. Embrechts, "Opening black box Data Mining models using Sensitivity Analysis," IEEE Symposium on Computational Intelligence and Data Mining (CIDM), 2011

[12] A. Jain and D. Zongker, "Feature selection: evaluation, application, and small sample performance", IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997, vol, 19(2), pp. 153-158.

[13] T. Tchaban, M. J. Taylor and J. P. Griffin, "Establishing Impacts of the Inputs in a Feedforward Neural Network," Neural Computing and Application, 1998, vol. 7, pp. 309-317

[14] A. Hunter, L. Kennedy, J. Henry and I. Ferguson, "Application of Neural Networks and Sensitivity Analysis to Improved Prediction of Trauma Survival," Computer Methods and Programs in Biomodicine, 2000, vol. 62, pp. 11-19.

[15] S. Lek, M. Delacoste, P. Baran, I. Dimopoulos, J. Lauga and S. Aulagnier, "Application of Neural Networks to Modeling NonLinear Relationships in Ecology," Ecological Modelling, 1996, vol. 90, pp. 39-52.

[16] Y. Dimopoulos, P. Bourret and S. Lek, "Use of Some Sensitivity Criteria for Choosing Neteworks with Good Generalization Ability," Neural Processing Letter, 1995, vol.2(6), pp. 1-4.

[17] I. Dimopoulos, J. Chronopoulos, A. Chronopoulou-Sereli and S. Lek, "Neural Network Models to Study Relationships between Lead Concentration in Grasses and Permanent Urban Descriptors in Athens City (Greece)," Ecological Modelling, 1999, vol. 120(2-3), pp. 157-165.

[18] G. D. Garson, "Interpreting Neural Network Connection Weights," Artificial Intelligence Expert, 1991, vol. 6, pp. 47-51.

[19] A. T. C. Goh, "Back-propagation Neural Networks for Modelling Complex Systems," Artificial Intelligence in Engineering, 1995, vol. 9, pp. 143-151.