A Quantum Approach Towards the Adaptive Prediction of Cloud Workloads

Ashutosh Kumar Singh, *Senior Member, IEEE*, Deepika Saxena, Jitendra Kumar, and Vrinda Gupta

Abstract—This work presents a novel Evolutionary Quantum Neural Network (EQNN) based workload prediction model for Cloud datacenter. It exploits the computational efficiency of quantum computing by encoding workload information into qubits and propagating this information through the network to estimate the workload or resource demands with enhanced accuracy proactively. The rotation and reverse rotation effects of the Controlled-NOT (C-NOT) gate serve activation function at the hidden and output layers to adjust the qubit weights. In addition, a Self Balanced Adaptive Differential Evolution (SB-ADE) algorithm is developed to optimize qubit network weights. The accuracy of the EQNN prediction model is extensively evaluated and compared with seven state-of-the-art methods using eight real world benchmark datasets of three different categories. Experimental results reveal that the use of the quantum approach to evolutionary neural network substantially improves the prediction accuracy up to 91.6 percent over the existing approaches.

Index Terms—Cloud computing, differential evolution, quantum neural network, workload forecasting

put to the end of the list general FL

1 Introduction

Cloud computing has emerged as an indispensable paradigm for computation and storage over the internet which is widely used in business, marketing, online transactions, research, academia, etc. [1]. It provides ubiquitous computing services benefit to all its users and freedom to pay-as-per-use model [2], [3]. Cloud service providers (CSP) utilize virtualization [4], [5] of physical resources at datacenters to maximize their revenue. The comprehensive working of datacenters depend on fine-grained provisioning of resources like storage, processor and network etc. [6], [7]. The workload demands show high variation over the time causing over/under-load and SLA violation problems where the static allocation of resources is unapplicable [8]. Evidently, there is a critical requirement of an efficient forecasting system that can predict upcoming resource utilization and workload demands with utmost precision to allow flawless functioning of cloud datacenters. The predicted workload information helps to analyze the right amount of resources to scale over a particular time interval, improve resource utilization, reliability (through prior prediction of system failures), minimize power consumption and make resource management decisions [2]. However, the accurate workload prediction is highly challenging due to the fluctuations in resource demands which has grabbed the attention of numerous researchers across the globe.

The classical workload forecasting methods based on time-series analysis, regression, and heuristic theories, artificial neural networks perform better for workloads with regular patterns. Meanwhile, the conventional neural network does not fully exploit the correlation between extracted patterns for accurate prediction [9]. It has been observed that quantum bits can draw more precise correlations from complex and dynamic input samples as compared to real numbers, which are used to set-up conventional neural network [10]. Hence, the learning capability of a conventional neural network can be enhanced by applying computational properties of quantum computing [11]. Quantum superposition functionality can be derived in the form of quantum states or qubit phase values as an operational unit by the quantum neural network (QNN) model [12]. Qubit phase values are generated from rotational effect, which have richer capability to learn more complex relationships during training process [10]. This gives the motivation for proposing a quantum approach based neural network prediction model for dynamic workload predictions. The most common method applied for training of conventional neural network is back propagation that employs single solution and tries to adjust it during learning process. However, it suffers from getting stuck into local minima and pre-mature convergence. On the other hand, evolutionary algorithms explore number of solutions to search an optimal solution and provide derivative-free optimization of neural network. As compared to GA and PSO, differential evolution (DE) has greater optimization capabilities in terms of convergence speed, computational complexity, accuracy and stability [13]. Therefore, a Self Balanced and Adaptive Differential Evolution (SB-ADE) algorithm is developed with self balancing capabilities of exploration and exploitation for optimization of proposed quantum approach based neural network prediction model. The key contributions are:

- *Ashutosh Kumar Singh and Deepika Saxena are with the Department of Computer Applications, NIT Kurukshetra, Haryana 136119, India. E-mail: ashutosh@nitkkr.ac.in, 13deepikasaxena@gmail.com.*
- *Jitendra Kumar is with the Department of Computer Applications, NIT Tiruchirappalli, Tamilnadu 620015, India. E-mail: jitendra@nitt.edu.*
- *Vrinda Gupta is with the Department of Electronics & Communication Engineering, NIT Kurukshetra, Haryana 136119, India. E-mail: vrindag16@nitkkr.ac.in.*

Manuscript received 6 Aug. 2020; revised 19 Apr. 2021; accepted 7 May 2021. Date of publication 11 May 2021; date of current version 3 June 2021. (Corresponding author: Deepika Saxena.) Recommended for acceptance by L. Y. Chen. Digital Object Identifier no. 10.1109/TPDS.2021.3079341

TABLE 1
Major Features: EQNN Versus State-of-the-Arts

Models	Training Algorithm	Datasets	Error metrics	AEF[a]	SA[b]
SaDE+NN [6]	Self adaptive DE	NASA and Saskatchewan	RMSE	×	×
RVLRBPNN [14]	Backpropagation	NASA	RMSE	×	×
AaDE+NN [15]	DE with three-phase adaptation	NASA and Saskatchewan	RMSE	×	✓
BaDE+NN [16]	Two-phase adaptive DE	NASA and Saskatchewan	RMSE	×	×
LSTM-RNN [17]	Backpropagation	Google Cluster and HPC/Grid System	MSE, MMSE	×	×
MLMVN [18]	Backpropagation	Sahara, Google Cluster	MSE	×	×
OM-FNN [19]	Variant of DE algorithm	Google Cluster	RMSE	×	×
EQNN	SB-ADE	Cluster:3, HPC:3, Web:2	Normalised RMSE, MAE	✓	✓

[a] *AEF:Absolute Error Frequency,* [b] *SA:Statistical Analysis.*

- A novel **E**volutionary **Q**uantum **N**eural **N**etwork (**EQNN**) based workload prediction model is proposed to proactively estimate the dynamic resource and job arrival demands with improved accuracy at cloud datacenters.
- SB-ADE learning algorithm is developed, which explores the search space globally and exploits the population to move closer towards a better solution to optimize the EQNN model.
- In-depth experimental analysis of the prediction accuracy of EQNN model is carried out over eight real world benchmark datasets belonging to three different categories. The comparison of EQNN prediction model with seven state-of-the-art methods confirms its superior accuracy.

The rest of the paper is organized as: Section 2 discusses the related work. The proposed approach is described in Section 3, which is divided into three subsections including, EQNN Prediction Model, its information processing, and training by SB-ADE algorithm. Section 4 entails performance evaluation followed by conclusive remarks and future scope of the proposed work in Section 5.

2 Related Work

The related work covers neural network based workload prediction and quantum inspired information processing.

2.1 Workload Prediction

A future workload prediction technique based on Back-propagation i.e., Random Variable Learning Rate Back-propagation Neural Network (RVLRBPNN) was developed in [14] to forecast the number of requests expected per prediction-interval. Later, Kumar *et al.* have proposed a self-adaptive differential evolution (SaDE) based artificial neural network model [6] for workload prediction which is capable of producing more accurate results as compared to RVLRBPNN gradient decent based backpropagation trained neural network model presented in [14]. Also, neural network based workload predictors trained with different evolutionary algorithms were presented in [15], [16], have outperformed the accuracy achieved by Back-propagation training based neural network. This was due to multi-directional searching with better exploration and learning capability of evolutionary algorithms over Back-propagation. The long short-term memory model in a recurrent neural network (LSTM-RNN) for fine-grained host load prediction was presented in [17]. Though the LSTM-RNN model learns long-term dependencies and produces high accuracy for the prediction of server loads, it suffers from long computation time during training due to the usage of Back-propagation algorithm between recurrent layers. To allow high capability of learning and better accuracy in less time, multi-layered neural networks with multi-valued neurons (MLMVN) prediction model was proposed in [18]. This approach used complex-valued neural network with a derivative-free feed-forward learning algorithm and produced better accuracy than LSTM-RNN approach. Recently, a neural network based multi-resource predictor is proposed in [19] which is trained using an adaptive evolutionary algorithm to predict multiple resources simultaneously. Table 1 compares major features of EQNN and state-of-the-art workload prediction approaches.

2.2 Quantum Based Information Processing

Many researchers have established that quantum computing can be assembled with machine learning through the concept of quantum parallelism [20], [21]. For instance, Quantum based fog scheduler is presented in [22] to determine optimal fog node for data analysis in real time. Quantum inspired information processing is based on qubits which allow faster learning and pattern recognition ability for complex relations between specific patterns of input data samples [23]. A quantum back propagation (QBP) neural network model was proposed in [24] for solving the 4-bit parity check problem that performed excellently in information processing efficiency than a classical neural network. Later, in [25], the learning capability of Qubit Neuron Network on 4-bit and 6-bit parity check problems have been investigated and results entailed that Qubit NN was far better than classical NN and complex NN that was due to the implication of superposition of quantum states at neuron node and the probability interpretation. Kouda *et al.* proposed multi-layer feed-forward QNN model [26] and analyzed it for high-level information processing, which revealed that QNN produced much better results than conventional Neural Network. The Qubit Multi-layer Perceptron (QuMLP) with a Quantum-Inspired Evolutionary Algorithm (QIEA) can search for the best specific time lags which is able to characterize the time series phenomenon, and can evolve the complete QuMLP architecture and parameters to control the random walk dilemma problem and has been analyzed for financial time series prediction in [27]. Cui *et al.* proposed a Complex Quantum Neuron (CQN) model for the time series prediction and the results improved due to realization of a deep quantum entanglement [28]. Ganjefar *et al.* [29] proposed a quantum states based neural

TABLE 2
Usage of QNN Across the Application Domain

Models	Underline Algorithm	Application	Performance metrics
Quantum computing inspired NN [22]	Gradient Descent	Prediction based Fog scheduler	task completion time and average energy consumption
Qubit NN [25]	Backpropagation	4-bit parity check problem	learning efficiency in terms of iterations
QuMLP [27]	QIEA	stock prediction	MSE, MAPE
CQN [28]	deep quantum entanglement	time series prediction	MSE, MAPE
Quantum inspired NN[29]	HGAGD	designing adaptive controller in power system	Integral Absolute Error, Integral Square Error, Integral Time Square Error

network optimized with hybrid genetic algorithm and gradient descent (HGAGD) studied the feasibility of designing an indirect adaptive controller for damping of low frequency oscillations in power systems and achieved satisfactory control performance. Table 2 summarises the operation and performance of QNN across the existing application domain.

Unlike existing works which have utilized traditional machine learning models and optimization algorithms for cloud workload prediction, the proposed work exploits the computational ability of quantum principles with machine learning capabilities of evolutionary neural network. Since the quantum systems can produce counter intuitive behavioural patterns, that are believed not to be efficiently produced by classical systems, it is reasonable to postulate that quantum computation combined with neural network optimization model outperforms the classical machine learning approaches in terms of its adaptability to forecast wider range of workloads with enhanced prediction accuracy.

3 Proposed Approach

The proposed workload prediction approach consists of three phases: *Learning*, *Testing*, and *Prediction* as shown in Fig. 1. During learning, the historical data is pre-processed by applying attribute extraction and normalization, followed by its division into training, validation, and testing data. The EQNN extracts suitable patterns from training data and finds out correlations among them to generate EQNN prediction trained model. The validation data is used to tune or optimize the model by evaluating its accuracy using an error evaluation function. If the desired accuracy is not achieved, the prediction model is retrained; otherwise, the model enters into the testing phase. During testing, the accuracy of the trained model is evaluated with unseen/test data to generate final prediction model. Finally, in load prediction phase, the EQNN prediction model is deployed for a proactive estimation of cloud workload and help in proficient load management. A detailed description of proposed approach is given in the subsequent sections.

3.1 EQNN Prediction Model

EQNN is an intelligent prediction model that utilizes the machine learning ability of evolutionary neural network and computational efficiency of quantum principles to maximize prediction accuracy. The fundamental unit of QNN is qubit,

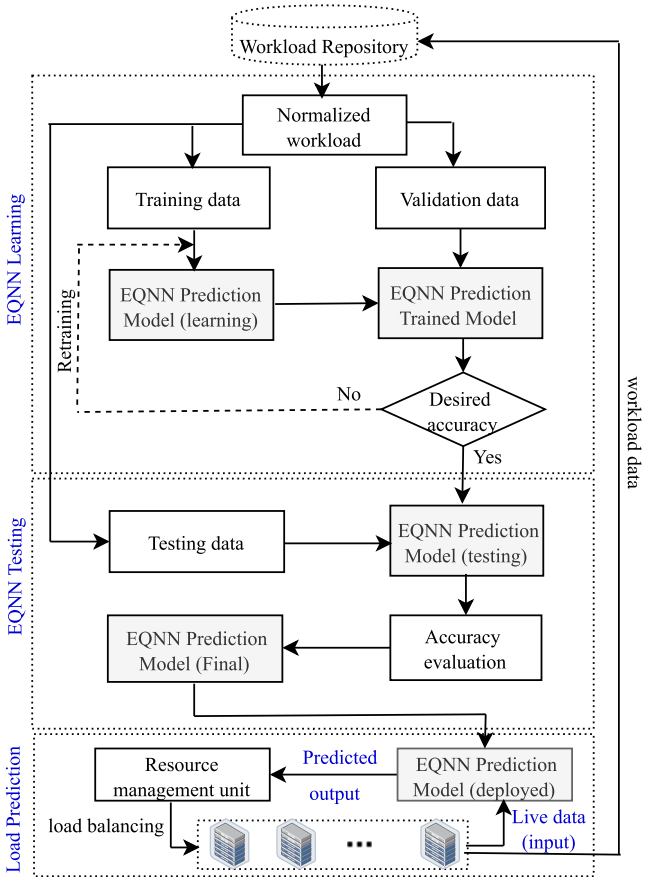


Fig. 1. Load prediction workflow.

which comprises of two-state quantum-mechanical system. A single qubit can represent a one, a zero, or crucially, any quantum superposition of these. Mathematically, a qubit can be realized as $|\Psi> = \alpha|0> + \beta|1>$, where α and β are complex numbers specifying the probability amplitudes of states $|0>$ and $|1>$ respectively.

The proposed EQNN prediction model shown in Fig. 2 consists of three layers, including input, hidden and output layers having n, p and q qubit nodes respectively which represents n-p-q qubit network architecture. The connection weights between qubit neurons of different layers are also taken in the form of qubits that are adjusted during training of EQNN. The qubit neuron state transitions are built upon the operations derived from quantum logic gates, including quantum rotation and C-NOT gates. The rotational effect of the quantum rotation gate is applied to generate qubits, and the rotation and reverse rotation functionality of the C-NOT gate is applied as an activation function at the hidden and output layers [30]. The quantum rotation or phase-shift gate transforms the phase of quantum state or qubit which is represented in Eq. (1)

$$R(\phi) = \begin{bmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{bmatrix}. \quad (1)$$

Consider initial quantum state $|\Psi> = |cos\phi_0, sin\phi_0|$, where $|\Psi>$ can be transformed by applying $R(\phi)$ that shifts the phase of $|\Psi>$ as depicted in Eq. (2).

$$R(\phi)|\Psi> = \begin{bmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{bmatrix} \begin{bmatrix} cos\phi_0 \\ sin\phi_0 \end{bmatrix} = \begin{bmatrix} cos(\phi+\phi_0) \\ sin(\phi+\phi_0) \end{bmatrix}. \quad (2)$$

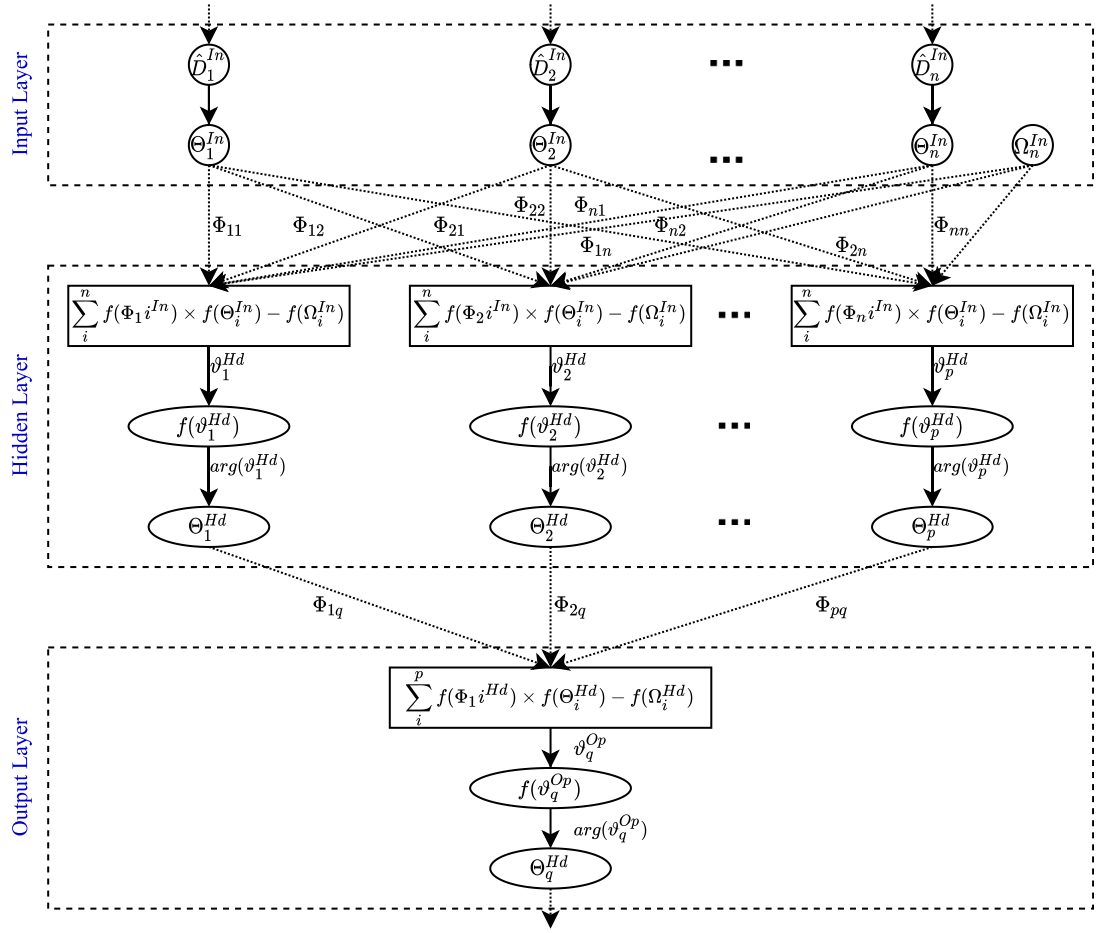


Fig. 2. Evolutionary quantum neural network (EQNN) model for workload prediction.

*Controlled-NOT gate*The C-NOT gate [25] performs reverse rotation and non-rotation of quantum states with respect to controlled input parameter η and can be defined as shown in Eq. (3), where $\eta = 1$ and $\eta = 0$ stands for reverse rotation and non-rotation, respectively.

$$f\Big(\frac{\pi}{2} \times \eta - \phi\Big) = \begin{cases} sin(\phi) + icos(\phi), & If(\eta = 1) \\ cos(\phi) - isin(\phi), & If(\eta = 0) \end{cases}. \quad (3)$$

3.1.1 Data Pre-Processing and Qubit Generation

The training input values (historical values) are extracted and aggregated into a specific time-interval (for example, 1 min, 5 min, 10 min and so on). The aggregation step is followed by the normalization of input data D in the range [0, 1] using $\hat{D} = \frac{D_i - D_{min}}{D_{max} - D_{min}}$, where D_{min} and D_{max} are the minimum and maximum values of the input data set, respectively. The normalized vector is denoted as $\hat{D}$, which is a set of all normalized input data values $\{\hat{D_1}, \hat{D_2}, \ldots, \hat{D_n}\}$. The normalized data is fed into the input layer of EQNN to transform the real-valued inputs into the quantum state phase values in the range $(-\pi/2, +\pi/2)$ by applying the effect of qubit rotation using ${y^{In}}_i = f({\Theta_i}^{In})$ and $\Theta_i = \frac{\pi}{2} \times d_i$, where d_i is the ith input data point, Θ_i is ith quantum input to the network.

These normalized values (in single dimension) are arranged as multi-dimensional qubit-input matrix and qubit-output matrix denoted as Θ_{input} and Θ_{output} respectively as stated in Eq. (4)

$$\Theta_{input} = \begin{bmatrix} \Theta_1 & \Theta_2 & \ldots. & \Theta_n \\ \Theta_2 & \Theta_3 & \ldots. & \Theta_{n+1} \\ . & . & \ldots. & . \\ . & . & \ldots. & . \\ \Theta_m & \Theta_{m+1} & \ldots. & \Theta_{n+m-1} \end{bmatrix} \Theta_{output} = \begin{bmatrix} \Theta_{n+1} \\ \Theta_{n+2} \\ . \\ . \\ \Theta_{n+m} \end{bmatrix}. \quad (4)$$

The QNN model extracts intuitive patterns from actual workload (Θ_{input}) and analyzes n previous workload values to predict the workload (Θ_{output}) about to arrive at next $(n+1)$th instance of time at the datacenter.

3.2 QNN Information Processing

The operational QNN model developed in the current approach is shown in Fig. 2. The Input layer neurons receive data as real valued data input $\{\hat{D_1}, \hat{D_2}, \ldots, \hat{D_n}\}$ and transform them into a set of qubit state vector such as $\{\Theta_1^{In}, \Theta_2^{In}, \ldots, \Theta_n^{In}\}$ by applying the steps mentioned in Qubit generation phase. In addition, a threshold qubit value (Ω^{In}) is generated in the range $[(-\pi/2, +\pi/2)]$ as a bias. The input layer is connected to the hidden layer

through qubit neural weights represented as Φ_{11}^{In}, $\Phi_{12}^{In}, \ldots, \Phi_{np}^{In}$.

At the Hidden layer, qubit vector such as $\{\Theta_1^{Hd}, \Theta_2^{Hd}, \ldots, \Theta_p^{Hd}\}$, is generated by computing $y_j^{Hd} = f(\Theta_j^{Hd})$ using Eqs. (5) and (6), where y_j^{Hd} is the jth output of hidden layer, and $f(\Theta_j^{Hd})$ states that an activation function is applied to Θ_j^{Hd}.

$$\Theta_j^{Hd} = \frac{\pi}{2} \times g(\delta^{Hd}) - arg(\vartheta_j^{Hd}) \tag{5}$$

$$\vartheta_j^{Hd} = \sum_{i=1,j=1}^{n,p} f(\Phi_{ij}^{In}) \times f(\Theta_i^{In}) - f(\Omega^{In}), \tag{6}$$

where Θ_j^{Hd} is jth (adjusted) qubit vector, obtained by applying reverse rotation or non-rotation effect of C-NOT gate, on amplitude (or magnitude) of qubit vector represented as $arg(\vartheta_j^{Hd})$, generated by applying activation function to ϑ_j^{Hd} obtained from Eq. (6). The expression $g(\delta^{Hd})$ is sigmoid function that produces value in the range [0, 1]. The Output layer generates prediction output as y^{Op}, which is derived by computing $y^{Op} = f(\Theta^{Op})$. Similar to the hidden layer, Θ^{Op} is a list of (adjusted) qubits, obtained by applying C-NOT gate effect on the amplitude of intermediate qubit vector i.e., $arg(\vartheta^{Op})$ as stated in Eq. (7). The intermediate qubit vector ϑ^{Op} can be obtained by applying Eq. (8).

$$\Theta^{Op} = \frac{\pi}{2} \times g(\delta^{Op}) - arg(\vartheta^{Op}) \tag{7}$$

$$\vartheta^{Op} = \sum_{i=1}^{p} f(\Phi_i^{Hd}) \times f(\Theta_i^{Hd}). \tag{8}$$

The $sigmoid()$ function shown in Eq. (9) imparts non-linearity and maps input to output in the range [0, 1]. Thereby, predicted qubit values are smoothen in the range [0, 1] to make them comparable with the actual output values. Finally, the performance and accuracy of the proposed model are evaluated by applying the root mean square error score (acting as fitness function) given in Eq. (10).

$$sigmoid(\varphi) = \frac{1}{1 + exp^{(-\varphi)}} \tag{9}$$

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(z_{actual} - z_{prediction})^2}. \tag{10}$$

3.3 Training by SB-ADE Algorithm

The training of EQNN begins with a randomly generated set of N qubit state vectors, where each vector represents a network of neural weight connections (computed using Eq. (11) and (12)), of size $(n+1) \times p + (p \times q) = p(n+q+1) \Leftarrow p(n+2)$, where $q = 1$ and additional 1 is added as bias at input layer.

$$\Psi = \alpha + i\beta \tag{11}$$

$$\Phi = phase(\Psi) \times \frac{\pi}{2}, \tag{12}$$

where $\alpha = sqrt(rd)$ and $\beta = sqrt(1 - rd)$ and $rd = random(0, 1)$. The term rd is a randomly generated number in range [0, 1], α and β are the probability amplitudes for realizing $|0>$ and $|1>$ respectively, Ψ is a quantum state in the Hilbert vector space and Φ is the phase angle determined for the corresponding qubit.

The number of maximum generations (epochs), mutation and crossover rate vectors are initialized within a specific range. Each network is evaluated on training data by applying a fitness function (Eq. (10)). The mutation strategy selection parameter (msp) is generated for each generation to select one of the four optional mutation schemes. The mutation and crossover operators are applied to generate offspring for next generation. The offspring vectors are evaluated by applying the fitness function which selects the optimal solutions for next generation. The mutation and crossover rates are updated after a fixed number of epochs ("learning period").

3.3.1 Mutation

SB-ADE optimization algorithm is developed with four different mutation schemes that balance the exploration and exploitation effects with adaptive generation of mutation learning rate (lp^M). The adoption of mutation strategy plays an important role in improving the quality of solutions. Each mutation strategy has some characteristics associated with it that justifies its specific usage. For example, $DE/best/1$ (Eq. (13)) and $DE/current-to-best/1$ (Eq. (14)) mutation strategies tend to be greedy as they use the best individual to generate mutant vectors which brings comprehensive exploitation during evolutionary stages of the algorithm. The mutation strategies $DE/current-to-rand/1/$ (Eq. (15)) and $DE/rand/1$ (Eq. (16)) help to find new search direction randomly that prevents premature convergence by allowing exploration [31] and raise population diversity that contributes to intensive exploration under control parameter κ. This will help in maintaining a good balance between exploration and exploitation properties of mutation variants.

$$\omega_i^j = \Phi_{best}^j + \mu_i \times (\Phi_{r1}^j - \Phi_{r2}^j) \tag{13}$$

$$\omega_i^j = \Phi_i^j + \mu_i \times (\Phi_{best}^j - \Phi_i^j) + \mu_i \times (\Phi_{r1}^j - \Phi_{r2}^j) \tag{14}$$

$$\omega_i^j = \Phi_i^j + \kappa_i \times (\Phi_{r1}^j - \Phi_i^j) + \mu_i \times (\Phi_{r2}^j - \Phi_{r3}^j) \tag{15}$$

$$\omega_i^j = \Phi_{r3}^j + \mu_i \times (\Phi_{r1}^j - \Phi_{r2}^j), \tag{16}$$

where ω_i^j and Φ_i^j depict ith mutant and current vectors respectively of jth iteration. The randomly generated numbers $r1$, $r2$ and $r3$ are mutually distinct numbers in the range [1, N]. The term Φ_{best}^j represents the best solution found so far, till jth generation. The parameters controlling mutation rate are μ_i and κ_i, for corresponding vector in ith iteration. Let Γ_1, Γ_2, Γ_3 and Γ_4 be the probabilities for opting the $DE/rand/1$, $DE/best/1$, $DE/current-to-best/1$ and $DE/current-to-rand/1$ mutation techniques, respectively. In reported experiments, initially Γ_1=Γ_2=

Γ_3=Γ_4= 0.25, so that each mutation scheme gets an equal chance of selection. Roulette wheel selection [32] scheme is utilized to choose among the mutation strategies that are based on the assigned probabilities. To implement the roulette wheel selection, a mutation selection probability (msp) vector of N random numbers in the range [0, 1], is generated. For example, if msp_i is less than or equal to Γ_1, $DE/rand/1$ is applied. Similarly, if the msp_i is greater than Γ_1 and smaller or equal to $\Gamma_1 + \Gamma_2$, $DE/best/1$ is applied, and so on. The mutation strategy selection is represented as $\wp$ in Eq. (17).

$$\wp = \begin{cases} \text{DE/current-to-best/1}, & If(0 < msp_i \leq \Gamma_1) \\ \text{DE/best/1}, & If(\Gamma_1 < msp_i \leq \Gamma_1 + \Gamma_2) \\ \text{DE/rand/1}, & If(\Gamma_1 + \Gamma_2 < msp_i \leq \Gamma_1 + \Gamma_2 + \Gamma_3) \\ \text{DE/current-to-rand/1}, & \text{otherwise} \end{cases}. \tag{17}$$

3.3.2 Crossover

The crossover operation is applied to mutant vector ω_i^j, and its current target vector Φ_i^j to produce new solutions called offspring χ_i^j i.e., ith solution of jth generation. A uniform crossover operation is applied to swap information at the gene level instead of segment level [33]. Eq. (18) shows a crossover operation where $\Re$ is a randomly generated number in the range [0, 1] for each gene in the chromosome. If the value of $\Re$ is smaller than the corresponding CR_{vi}^j, then the crossover is successfully applied to exchange ith gene of the mutant vector with the current target vector.

$$\chi_i^j = \begin{cases} \omega_i^j & If(\Re \in (0,1) \leq CR_{vi}^j) \\ \Phi_i^j & \text{otherwise.} \end{cases} \tag{18}$$

In the proposed approach, the crossover rate is initialized randomly in the range [0, 1] with mean value CR_μ as 0.5, and standard deviation CR_σ as 0.1. The crossover rate gets update after a fixed number of generations (known as crossover learning period) with new values of the mean and standard deviation of crossover rate which are recorded during evolutionary process.

During each generation or epoch, the number of candidates successfully reaching the next generation denoted as ξ_1, ξ_2, ξ_3, and ξ_4, for four different mutation strategies, are monitored. Similarly, Δ_1, Δ_2, Δ_3, and Δ_4 record the number of candidates who failed to reach the next generation. The probabilities for opting $DE/rand/1$, $DE/best/1$, $DE/current-to-best/1$, and $DE/current-to-rand/1$ mutation strategies are computed as Γ_1, Γ_2, Γ_3, and Γ_4 by applying Eq. (19).

$$\begin{aligned} Z &= 2(\xi_2\xi_3\xi_4 + \xi_1\xi_3\xi_4 + \xi_1\xi_3\xi_2 + \xi_2\xi_3\xi_4) \\ &\quad + \Delta_1(\xi_2 + \xi_3 + \xi_4) + \Delta_2(\xi_1 + \xi_3 + \xi_4) \\ &\quad + \Delta_3(\xi_1 + \xi_2 + \xi_4) + \Delta_4(\xi_1 + \xi_3 + \xi_2) \\ \Gamma_2 &= \frac{\xi_2(\xi_1 + \Delta_1 + \xi_3 + \Delta_3 + \xi_4 + \Delta_4)}{Z} \\ \Gamma_3 &= \frac{\xi_3(\xi_1 + \Delta_1 + \xi_2 + \Delta_2 + \xi_4 + \Delta_4)}{Z} \\ \Gamma_4 &= 1 - (\Gamma_1 + \Gamma_2 + \Gamma_3). \end{aligned} \tag{19}$$

3.3.3 Selection

The population for the next generation is selected by applying survival of the fittest concept using Eq. (20) where Φ_i^{j+1} is the selected candidate for next generation, χ_i^j is the solution generated after crossover, and Θ_i^j is currently existing candidate solution.

$$\Phi_i^{j+1} = \begin{cases} \chi_i^j & If(fitness(\chi_i^j) \leq (fitness(\Theta_i^j)) \\ \Phi_i^j & \text{otherwise.} \end{cases} \tag{20}$$

3.4 Algorithm and Time Complexity

Algorithm 1 gives an operational summary of EQNN optimization. The time complexity of SB-ADE depends on the number of networks (N), number of network weight connections (D), number of input nodes (n), hidden nodes (p), where $p \simeq n$. Hence, the total time complexity for a maximum M number of generations comes out to be $M \times O(n^2 \times N \times D) \Rightarrow O(n^2 dNM)$.

Algorithm 1. EQNN Training Algorithm

```
Initialize CR_μ = F_μ = 0.5, CR_σ = 0.1, F_σ = 0.3, maximum generations (M), Γ1=Γ2=Γ3=Γ4=0.25;
Number of weight connection in each network D = (n+1)×p + (p×q) = p(n+q+1) ⇒ p(n+2) for q=1;
Initialize N networks of length D randomly;
Generate CR_v and M_v of size N, CR_v = (CR_μ, CR_σ, D), F_v = (F_μ, F_σ, N);
Evaluate each network on training data using fitness function;
for each generation i ≤ M do
  Generate vector msp of N random number ∈ [0,1];
  for each network j ≤ N do
    Generate r1 ≠ r2 ≠ r3 ≠ r4 ≠ i ∈ [1, N] and K_rand ∈ [1, D];
    if 0 < msp_i ≤ Γ1 then
      Apply DE/rand/1 (Eq. (13));
    else if msp_i ≤ (Γ1 + Γ2) then
      Apply DE/best/1 (Eq. (14));
    else if msp_i ≤ (Γ1 + Γ2 + Γ3) then
      Apply DE/current − to − best/1 (Eq. (15));
    else
      Apply DE/random/1 (Eq. (16));
    end
    Apply uniform crossover;
  end
  Evaluate newly generated solutions using error estimation function;
  Select participants for next generation using Eq. (20);
  Update values of ξ1, ξ2, ξ3, ξ4, Δ1, Δ2, Δ3, Δ4;
  Update Γ1, Γ2, Γ3, Γ4 after lp^M generations;
  Update CR_v after lp^CR generations;
end
```

4 Performance Evaluation

4.1 Experimental Set-up

The simulation experiments are conducted on a server machine assembled with two Intel® Xeon® Silver 4114 CPU with a 40 core processor and 2.20 GHz clock speed. The

TABLE 3
Experimental Set-up Parameters and Their Values

Parameter	Value
Input neural nodes (n)	7-28
Hidden layer nodes (p)	4-20
Output layer nodes(q)	1
Maximum epochs(G_{max})	250
Size of training data	75%
Mutation learning period(lp^M)	10
Crossover rate learning period(lp^{CR})	10
Number of population	15

TABLE 4
Characteristics of Evaluated Workloads From Cluster, Web, and HPC Applications

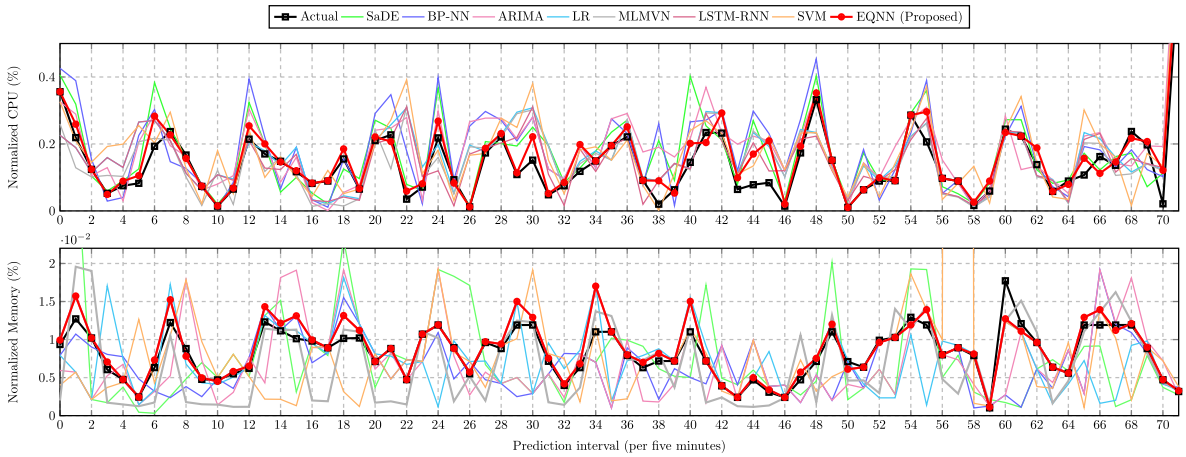
Category	Workload	Duration	Jobs	Mean(%)	St.dev
Cluster	GCD-CPU	10 days	2 M	21.84	13.62
	GCD-memory	10 days	2 M	19.55	16.6
	PL-CPU	10 days	1.5 M	19.77	14.55
Web	NASA-HTTP	31 days	1.8 M	2.95	1.72
	Saskatchewan	7 months	2.5 M	2.97	1.73
HPC	AuverGrid	365 days	2.3 M	1.27E+9	4.18E+4
	NorduGrid	60 days	122K	9.56E+4	4.64E+5
	SHARCNet	11 days	188K	1.14E+9	4.89E+5

computation machine is deployed with 64-bit Ubuntu 16.04 LTS, having 128 GB RAM. The proposed work is implemented in Python version 3.7 with the details listed in Table 3.

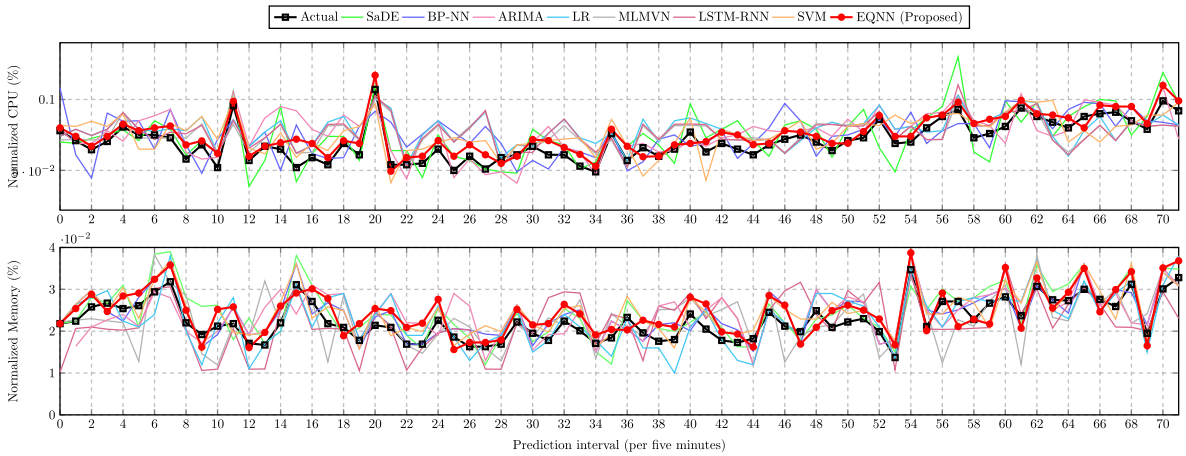
4.2 Data Sets

Three different categories of benchmark datasets have been chosen for thorough analysis of the proposed work including Cluster workload from Google Cluster Data (GCD) [34] and PlanetLab Virtual Machines (VMs) traces [35], Web servers workload collected from NASA and Saskatchewan web server request traces [36] and High Performance Computing (HPC) grid workloads belonging to Grid workload Archive [37]. GCD and Planet Lab (PL) workloads entail behavior of cloud applications for cluster and big data analytics (e.g., Hadoop) which gives resource usage traces collected over a period of 29 days in May, 2011. The CPU and memory utilization are extracted and aggregated over the period of first ten days for different prediction intervals like, 5 min, 10 min, ..., 60 min. The Planet Lab dataset contains 11,746 VMs traces which are measured at interval of five minutes and collected in March and April, 2011.

NASA (NS) and Saskatchewan (SK.) HTTP traces contain information of Host, Time-stamp, HTTP request,



(a) VM 1 (Periodically variable load)



(b) VM 2 (Randomly variable load)

Fig. 3. Actual versus predicted CPU and memory utilization for two selected Google cluster VMs.

TABLE 5
RMSE and Execution Time Elapsed for Different Combinations of Input Nodes and Hidden Nodes

Input nodes	Hidden nodes	RMSE	Training time (msec)	Iterations
7	4	0.0076	116.67	17
10	7	0.0022	234.89	20
15	10	0.0281	249.94	22
20	14	0.0430	533.70	19
25	18	0.0615	618.11	22

TABLE 6
RMSE (10^{-3}) for Cluster Workloads (DS:Datasets)

DS	PWS (min)	SVM	LR	ARIMA	BP	SaDE	MLM -VN	LSTM -RNN	EQNN
GCD-CPU	5	1.42	1.16	1.11	1.25	1.68	1.97	1.23	0.89
GCD-Mem		6.18	6.01	5.12	2.48	4.08	3.61	1.91	1.80
PL-CPU		8.41	7.30	8.51	7.43	2.30	2.98	3.10	2.11
GCD-CPU	30	9.40	8.84	11.9	27.9	12.3	4.24	1.30	1.14
GCD-Mem		9.41	8.85	1.70	20.8	6.78	6.21	9.13	1.33
PL-CPU		16.6	15.3	12.3	25.6	15.3	53.1	20.4	14.6
GCD-CPU	60	9.46	10.17	11.7	17.9	34.9	14.7	10.2	2.92
GCD-Mem		10.5	12.02	13.50	30.6	8.92	12.6	11.3	9.60
PL-CPU		6.14	5.36	4.70	20.8	10.6	12.2	18.7	9.21

HTTP reply and Bytes sent in the reply in ASCII files. For experimental analysis, we have extracted the Time-stamp values from NASA and Saskatchewan web server logs, which are aggregated into different prediction intervals, and the resultant values are normalized. Many scientific applications actively use cloud environment to assist diverse scientific computing and achieve high performance. The proposed EQNN prediction model is evaluated on three HPC grid workloads including AuverGrid (AG) [38], NorduGrid (NG) [39], and SHARCNet (SCN) [40]. Table 4 shows the statistical characteristics of the evaluated workloads.

4.3 Prediction Accuracy Analysis and Comparison

The proposed EQNN prediction model is compared with widely used Support Vector Machine (SVM) [41], Linear Regression (LR) [42], Autoregressive integrated moving average (ARIMA) [43], [44], Neural Network (NN) based on Backpropagation (BPNN) [14], neural network optimized with Self-adaptive Differential Evolution (SaDE) algorithm [6], MLMVN [18], LSTM-RNN [17] and Deep Learning [45] workload prediction models. The accuracy of EQNN prediction model is thoroughly investigated on a wide variety of cloud applications, variable workloads and estimation of resource utilization by VMs. Fig. 3 compares actual and predicted resources (CPU and memory) utilization using EQNN and other comparative methods for a periodically and randomly variable workload by VM1 (Fig. 3a) and VM2 (Fig. 3b) respectively. The proposed method estimate the CPU and memory utilization closer to the actual resource utilization for both VMs serving a significantly different type of workload. Moreover, it is difficult to forecast in the presence of random peaks, as shown in the given Fig. 3, which are precisely estimated by the proposed method. In addition, the performance of the proposed work is analyzed for a different number of input and hidden neurons, where the most efficient outcome was obtained with ten input neurons as reported in Table 5. The number of hidden nodes were chosen to be seven [6]. It has been noticed during experimental analysis that the

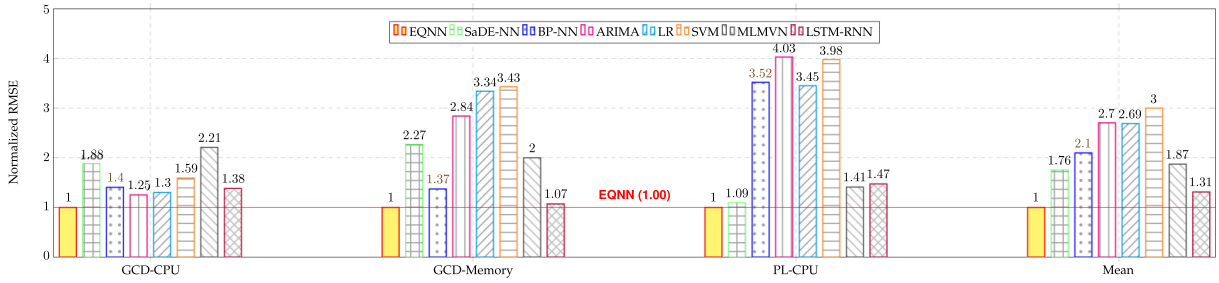
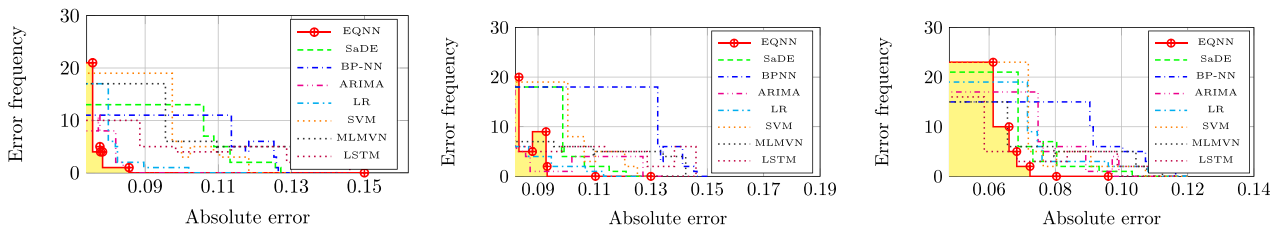


Fig. 4. Normalized RMSE results in cluster workloads.



(a) GCD-CPU (b) GCD-Memory (c) PL-CPU

Fig. 5. Absolute error frequency of resource utilization estimation for cluster workloads.

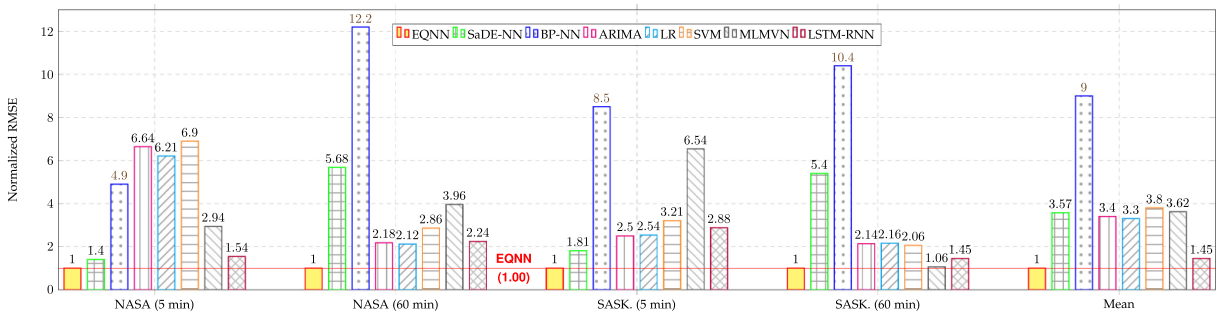
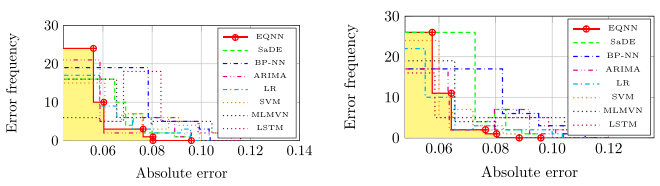


Fig. 6. Normalized RMSE results in Web workloads.

performance of the model degrades as the number of input neurons increases beyond 10. The reason is that as the network size grows up, it starts memorizing the training set and not correlating or learning the patterns. On the other hand, with a smaller network size, it underestimates the useful information from the training set and shows a lesser ability to develop necessary correlations that result in performance degradation with reduced pattern learning. The subsequent sections entail the accuracy comparison for different workloads.

4.3.1 With Cluster Workloads

Fig. 4 shows the normalized RMSE results of predicted resource utilization by applying EQNN and other seven comparative approaches for the three cluster workloads, where 1.00 indicates the result of proposed EQNN prediction model and higher values indicate worse performance. All the RMSE results are normalized with respect to RMSE output of EQNN. Also, the prediction error obtained by each method is quantified and visualized by computing absolute prediction error and analyzing its frequency for all three workloads. Fig. 5 shows the frequency of absolute error (*Actual value - Predicted value*), where EQNN yields minimum error for the majority of the workloads as compared to other seven methods. Table 6 presents RMSE value obtained for various prediction intervals including 5 min, 30 min and 60 minutes, where PWS is Prediction Window Size. The least error is obtained for 5 minutes prediction interval i.e., 0.00089 and 0.0018 for CPU and memory demand traces, respectively. It is found that RMSE increases with the size of prediction interval due to decrement in the number of data samples. On average, EQNN is 6-91.6 percent more accurate than the seven comparative methods. This is because qubits have more diversity than real-valued weights and allow enhanced intuitive learning of the patterns during training process to precisely predict the unseen future variations.



(a) NASA HTTP traces (b) Saskatchewan HTTP traces

Fig. 7. Absolute error frequency.

4.3.2 With Web Workloads

Fig. 6 shows the normalized RMSE bar plots of actual and predicted workload for job arrival during two prediction intervals viz. 5 minutes and 60 minutes using EQNN and other seven comparative approaches for NASA and Saskatchewan HTTP traces. The resulted plots clearly depict that EQNN improves prediction accuracy by more than 90 percent against all the seven methods for each prediction interval in case of web workloads. The frequency of absolute error for prediction results of both NASA and Saskatchewan HTTP traces are plotted in Fig. 7, where proposed EQNN always delivers the least prediction error as compared to other comparative approaches. Moreover, the high frequency of absolute error in the case of EQNN signifies the robustness and strong tendency for yielding maximum prediction accuracy. Table 7 shows comparative prediction accuracy of existing and proposed EQNN prediction model over different web workloads for different prediction intervals, where least prediction error is obtained with Prediction Window Size (PWS) of five minutes, and thereafter prediction error increases slightly due to decrease in the number of training data samples for bigger PWS. The reason for such an improved accuracy of predicted values is thorough learning of useful information by QNN model from input data samples. The C-NOT gate allows a rotational effect that traces the input data intensively and generates counter-intuitive patterns from it, and closely anticipates the future workload information.

4.3.3 With HPC Workloads

The normalized RMSE results of predicted resource utilization by applying EQNN and other seven comparative approaches for the three scientific or high performance

TABLE 7
RMSE (10^{-3}) for Web Workloads

DS	PWS (min)	SVM	LR	ARIMA	BP	SaDE	MLM -VN	LSTM -RNN	EQNN
NS	5	4.92	4.41	4.72	3.51	1.00	2.09	1.10	0.71
SK.		3.54	2.86	2.75	9.36	2.00	7.2	3.17	1.10
NS	30	5.97	4.71	4.30	29.2	14.2	6.91	6.43	2.50
SK.		4.82	4.53	4.31	34.9	20.9	26.5	4.79	3.50
NS	60	7.15	5.31	5.47	30.5	14.2	9.91	5.59	2.50
SK.		6.40	6.70	6.66	32.3	17.0	3.31	4.50	3.10

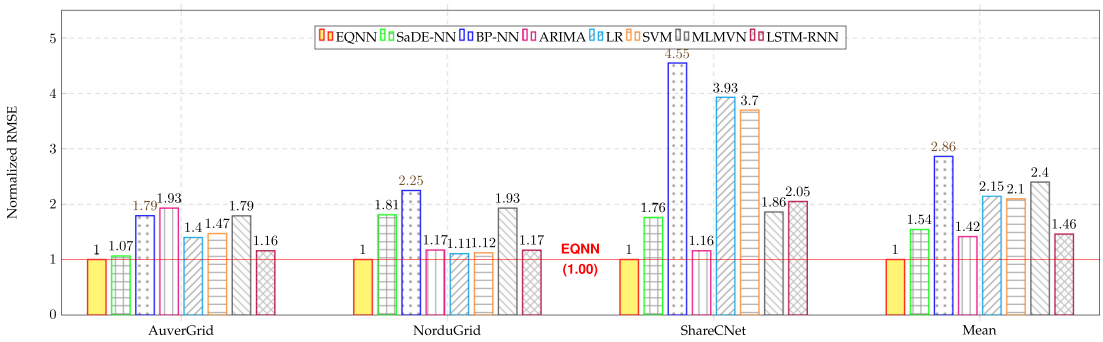
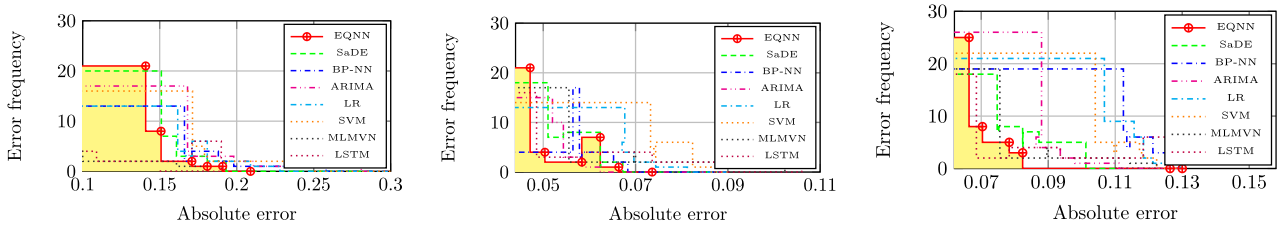


Fig. 8. Normalized RMSE results in HPC workloads.



(a) AuverGrid (b) NorduGrid (c) ShareCNet

Fig. 9. Absolute error frequency of number of jobs estimation for HPC workloads.

computing workloads are shown in Fig. 8. On average, EQNN achieves prediction accuracy improvement of 25 - 96 percent over the seven comparative methods. Fig. 9 shows the frequency of absolute error, where EQNN yield minimum error for the majority of the workloads as compared to other seven methods. Table 8 presents RMSE value obtained for various prediction intervals including 5 min, 30 min, and 60 minutes. The forecasting accuracy is improved due to the usage of qubits in the form of phase values which draws counter intuitive patterns based on behavior of the application and allows extensive exploration and better learning by neural network as compared to real valued neural network. Moreover, the application of an optimally adaptive differential evolution algorithm for training of EQNN prediction model with effective balancing of exploration and exploitation has enhanced the learning capabilities of prediction model. It allows searching in multiple directions and applies a greedy approach as well as raises diversity of the search space to find out optimal/ near-optimal solution. Additionally, the proposed work

TABLE 8
RMSE (10^{-3}) Over Scientific/HPC Grid Workloads

DS	PWS (min)	SVM	LR	ARIMA	BP	SaDE	MLM -VN	LSTM -RNN	EQNN
AuverG.	5 min	4.45	4.23	5.83	5.42	3.21	5.42	3.52	3.02
NorduG.		2.39	2.36	2.50	4.80	3.89	4.12	3.05	2.13
SHARCNet		9.6	10.2	3.00	11.8	4.56	4.82	5.3	2.59
AuverG.	30 min	7.78	9.38	9.10	9.52	8.92	12.2	8.95	6.71
NorduG.		32.5	28.5	43.5	48.2	13.9	18.9	32.2	10.2
SHARCNet		5.78	4.10	20.0	6.76	8.93	11.8	9.92	4.79
AuverG.	60 min	269.6	272.6	22.4	17.8	20.1	9.2	14.0	10.6
NorduG.		13.8	14.7	19.2	22.9	18.3	25.6	17.9	11.2
SHARCNet		25.2	33.9	10.6	25.4	12.2	22.4	14.4	6.35

TABLE 9
MAE Comparison on GCD-CPU Dataset

Approach	ARIMA [43]	SVM [43]	Proposed
5 min	0.9720	0.9690	0.3715
60 min	0.9210	-	0.4827

TABLE 10
RMSE (10^{-3}) Comparison on Planet Lab CPU

Approach	5 min	30 min	60 min
Deep learning [45]	9.1700	10.3000	9.9700
Proposed	0.0021	0.0146	0.00921

TABLE 11
Training-Time (sec) for Prediction-Intervals

Approach	5 min	30 min	60 min
SVM	1.08	0.50	0.25
LR	0.84	0.38	0.24
BP	24.11	4.27	2.91
SaDE	48.58	4.27	2.91
ARIMA	1769.07	75.18	29.05
MLMVN	179.11	36.10	19.23
LSTM-RNN	193.07	39.28	16.45
Proposed (EQNN)	223.41	26.01	11.91

TABLE 12
Friedman Test Result

Dataset	Statistic	p-value	Result
GCD-CPU	2.92	0.07009	H_0 is accepted
GCD-Mem.	2.26	0.12803	H_0 is accepted
PL-CPU	12.00	0.000058	H_0 is rejected
NASA HTTP	822.27	0.000055	H_0 is rejected
Saskatchewan HTTP	322.27	0.00069	H_0 is rejected
AuverGrid	2.92	0.07009	H_0 is accepted
NorduGrid	11.13	0.000079	H_0 is rejected
SHARCNet	2.37	0.11452	H_0 is accepted

is compared with prediction results achieved in ARIMA and SVM [43] on google cluster CPU traces by applying Mean Absolute Error (MAE) for PWS of 5 and 60 minutes as shown in Table 9, where the – represents unavailability of the corresponding result. The RMSE results of EQNN and Deep learning [45] obtained on Planet Lab CPU traces are also compared in Table 10. The comparison of training-time of the proposed and comparative approaches is shown in Table 11, which is highest for ARIMA, followed by the proposed approach (due to generation of qubit-based network weights) and least for LR. However, the efficiency and applicability of the proposed predictor is not affected because the training is a periodic task and can be executed in parallel on the servers equipped with enough resources.

4.4 Statistical Analysis

The obtained results are validated by conducting a statistical test on STAC [46] web platform and applying the Friedman test followed by Finner post hoc analysis. The Friedman test considers a null hypothesis (H_0) which assumes that there is no significant difference in the results of different approaches. Finner post hoc analysis analyzes the pairwise performance of the algorithms. The tests are conducted by using SB-ADE algorithm as a control method with a significance level of 0.05. Table 12 shows that the Friedman test rejects the H_0 for NASA, Saskatchewan HTTP traces, Planet Lab CPU and NorduGrid workload indicates the presence of a significant difference in the results. However, it is accepted for Google Cluster CPU and memory; AuverGrid and SHARCNet specify the absence of this difference. The ranks obtained by different approaches through the Friedman test are shown in Table 13. It can be observed that the best rank is obtained by the proposed method in majority of the

TABLE 13
Friedman Test Ranks

Approach	GCD-CPU	GCD-Mem.	PL-CPU	NASA	Saskatchewan	AuverGrig	NorduGrid	SHARCNet
EQNN	1.6667	1.3333	3.6667	1.0000	1.0833	1.0000	1.0000	1.3333
SaDE	5.3333	2.3333	4.6667	2.9166	2.5833	2.6667	3.6667	3.6667
ARIMA	3.3333	4.0000	1.3333	2.0833	2.3333	4.6667	4.6667	4.3333
LR	2.33333	4.0000	1.6667	3.6667	3.3333	4.6667	2.6667	2.3333
SVM	3.3333	4.6667	3.8333	3.3333	3.6667	3.6667	3.0000	4.6667
BP-NN	5.0000	4.6667	5.8333	4.0000	4.166	4.3333	6.0000	4.6667

TABLE 14
Finner Post-Hoc Analysis Result

Comparison	NASA HTTP			Saskatchewan HTTP		
	Statistics	Adjusted p-value	H_0	Statistics	Adjusted p-value	H_0
EQNN vs SVM	4.38	0.00005	rejected	4.10	0.00016	rejected
EQNN vs LR	2.27	0.0060	rejected	2.37	0.0014	rejected
EQNN vs ARIMA	1.18	0.23533	accepted	2.16	0.03038	rejected
EQNN vs BPNN	3.28	0.0020	rejected	3.37	0.0014	rejected
EQNN vs SaDE	2.09	0.0047	rejected	2.59	0.01866	rejected
Comparison	**GCD-CPU**			**PL-CPU**		
	Statistics	Adjusted p-value	H_0	Statistics	Adjusted p-value	H_0
EQNN vs SVM	1.09	0.4152	accepted	0.109	0.91312	accepted
EQNN vs LR	0.44	0.4364	accepted	1.3093	0.49185	accepted
EQNN vs ARIMA	1.09	0.4152	accepted	1.52759	0.49185	accepted
EQNN vs BPNN	2.18	0.0792	accepted	1.4184	0.49185	accepted
EQNN vs SaDE	2.40	0.0792	accepted	0.65465	0.59285	accepted
Comparison	**AuverGrid**			**NorduGrid**		
	Statistics	Adjusted p-value	H_0	Statistics	Adjusted p-value	H_0
EQNN vs SVM	1.75	0.10003	accepted	1.30	0.23208	accepted
EQNN vs LR	2.40	0.07925	accepted	1.09	0.27523	accepted
EQNN vs ARIMA	2.40	0.07925	accepted	2.40	0.04044	rejected
EQNN vs BPNN	2.18	0.07925	accepted	3.27	0.00530	rejected
EQNN vs SaDE	1.09	0.27523	accepted	1.74	0.13109	rejected

cases. The Finner analysis given in Table 14 shows a pair-wise comparison of presence of significant difference between the two approaches where the comparison of the pairs are rejected and accepted as well which indicates the presence of a significant difference and no difference between them, respectively. Hence, the proposed EQNN based prediction approach outperforms the state-of-the-art prediction methods, and it is suitable to enhance the accuracy of workload forecasting at the cloud datacenter.

5 Conclusion and Future Work

This work proposed a novel EQNN model to forecast future workload demands at the cloud datacenter. The input data points and neural weight connections are qubit phase values. A self-balanced adaptive differential evolution (SB-ADE) algorithm is developed to train this model. The experimental analysis of the proposed work is done by using eight benchmark real workload datasets, and the EQNN model surpasses the prediction accuracy over state-of-the-art approaches. The future aim of this work would be to raise the quality of prediction further and allow online training with live data for prediction of bandwidth requirement and power consumption in future at cloud datacenters.

Acknowledgments

The authors would like to thank National Institute of Technology, Kurukshetra, India for financially supporting the research work. Deepika Saxena and Ashutosh Kumar Singh are co-first authors.

References

[1] R. Buyya *et al.*, "A manifesto for future generation cloud computing: Research directions for the next decade," *ACM Comput. Surv.*, vol. 51, no. 5, 2018, Art. no. 105.

[2] D. Saxena, A. K. Singh, and R. Buyya, "OP-MLB: An online VM prediction based multi-objective load balancing framework for resource management at cloud datacenter," *IEEE Trans. Cloud Comput.*, early access, doi: 10.1109/TCC.2021.3059096.

[3] D. Saxena and A. K. Singh, "Energy aware resource efficient-(EARE) server consolidation framework for cloud datacenter," in *Proc. Adv. Commun. Comput. Technol.*, 2020, pp. 1455–1464.

[4] W. Song, Z. Xiao, Q. Chen, and H. Luo, "Adaptive resource provisioning for the cloud using online bin packing," *IEEE Trans. Comput.*, vol. 63, no. 11, pp. 2647–2660, Nov. 2014.

[5] D. Saxena and A. Singh, "Security embedded dynamic resource allocation model for cloud data centre," *Electron. Lett.*, vol. 56, no. 20, pp. 1062–1065, 2020.

[6] J. Kumar and A. K. Singh, "Workload prediction in cloud using artificial neural network and adaptive differential evolution," *Future Gener. Comput. Syst.*, vol. 81, pp. 41–52, 2018.

[7] A. K. Singh and D. Saxena, "A cryptography and machine learning based authentication for secure data-sharing in federated cloud services environment," *J. Appl. Secur. Res.*, pp. 1–24, 2021.

[8] J. Bi, H. Yuan, and M. Zhou, "Temporal prediction of multiapplication consolidated workloads in distributed clouds," *IEEE Trans. Automat. Sci. Eng.*, vol. 16, no. 4, pp. 1763–1773, Oct. 2019.

[9] Z. Chen, J. Hu, G. Min, A. Y. Zomaya, and T. El-Ghazawi, "Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 4, pp. 923–934, Apr. 2020.

[10] A. A. Ezhov and G. Berman, "Role of interference and entanglement in quantum neural processing," in *Proc. Electron. Structures MEMS II*, 2001, pp. 367–379.

[11] D. Cutting, "Would quantum neural networks be subject to the decidability constraints of the church-turing thesis," *Neural Netw. World*, vol. 9, pp. 163–168, 1999.

[12] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[13] B. Hegerty, C.-C. Hung, and K. Kasprak, "A comparative study on differential evolution and genetic algorithms for some combinatorial problems," in *Proc. 8th Mex. Int. Conf. Artif. Intell.*, 2010, pp. 177–186.

[14] Y. Lu, J. Panneerselvam, L. Liu, and Y. Wu, "RVLBPNN: A workload forecasting model for smart cloud computing," *Sci. Program.*, vol. 2016, pp. 1–9, 2016.

[15] D. Saxena and A. K. Singh, "Auto-adaptive learning-based workload forecasting in dynamic cloud environment," *Int. J. Comput. Appl.*, vol. 42, pp. 1–11, 2020.

[16] J. Kumar, D. Saxena, A. K. Singh, and A. Mohan, "Biphase adaptive learning-based neural network model for cloud datacenter workload forecasting," *Soft Comput.*, vol. 24, no. 1, pp. 1–18, 2020.

[17] J. Kumar, R. Goomer, and A. K. Singh, "Long short term memory recurrent neural network (LSTM-RNN) based workload forecasting model for cloud datacenters," *Procedia Comput. Sci.*, vol. 125, pp. 676–682, 2018.

[18] K. Qazi and I. Aizenberg, "Cloud datacenter workload prediction using complex-valued neural networks," in *Proc. IEEE 2nd Int. Conf. Data Stream Mining Process.*, 2018, pp. 315–321.

[19] D. Saxena and A. K. Singh, "A proactive autoscaling and energy-efficient VM allocation framework using online multi-resource neural network for cloud data center," *Neurocomputing*, vol. 426, pp. 248–264, 2020.

[20] C. Havenstein, D. Thomas, and S. Chandrasekaran, "Comparisons of performance between quantum and classical machine learning," *SMU Data Sci. Rev.*, vol. 1, no. 4, 2018, Art. no. 11.

[21] S. Imre, "Quantum existence testing and its application for finding extreme values in unsorted databases," *IEEE Trans. Comput.*, vol. 56, no. 5, pp. 706–710, May 2007.

[22] M. Bhatia, S. K. Sood, and S. Kaur, "Quantum-based predictive fog scheduler for IoT applications," *Comput. Ind.*, vol. 111, pp. 51–67, 2019.

[23] R. Zhou and Q. Ding, "Quantum MP neural network," *Int. J. Theoretical Phys.*, vol. 46, no. 12, pp. 3209–3215, 2007.

[24] N. Matsui, N. Kouda, and H. Nishimura, "Neural network based on QBP and its performance," in *Proc. Int. Joint Conf. Neural Netw. Neural Comput., New Challenges Perspectives New Millennium*, 2000, pp. 247–252.

[25] N. Kouda, N. Matsui, H. Nishimura, and F. Peper, "Qubit neural network and its learning efficiency," *Neural Comput. Appl.*, vol. 14, no. 2, pp. 114–121, 2005.

[26] N. Kouda, N. Matsui, and H. Nishimura, "A multilayered feed-forward network based on qubit neuron model," *Syst. Comput. Japan*, vol. 35, no. 13, pp. 43–51, 2004.

[27] R. d. A. Araújo, A. L. de Oliveira, and S. C. Soares, "A quantum-inspired hybrid methodology for financial time series prediction," in *Proc. Int. Joint Conf. Neural Netw.*, 2010, pp. 1–8.

[28] Y. Cui, J. Shi, and Z. Wang, "Complex rotation quantum dynamic neural networks (CRQDNN) using complex quantum neuron (CQN): Applications to time series prediction," *Neural Netw.*, vol. 71, pp. 11–26, 2015.

[29] S. Ganjefar and M. Tofighi, "Training qubit neural network with hybrid genetic algorithm and gradient descent for indirect adaptive controller design," *Eng. Appl. Artif. Intell.*, vol. 65, pp. 346–360, 2017.

[30] C. P. dos Santos Gonçalves, "Quantum neural machine learning: Theory and experiments," in *Machine Learning in Medicine and Biology*. London, UK: IntechOpen, pp. 95–115, 2019.

[31] A. W. Iorio and X. Li, "Solving rotated multi-objective optimization problems using differential evolution," in *Proc. Australas. Joint Conf. Artif. Intell.*, 2004, pp. 861–872.

[32] L. Zhang, H. Chang, and R. Xu, "Equal-width partitioning roulette wheel selection in genetic algorithm," in *Proc. Conf. Technol. Appl. Artifi. Intell.*, 2012, pp. 62–67.

[33] G. Pavai and T. Geetha, "A survey on crossover operators," *ACM Comput. Surv.*, vol. 49, no. 4, 2017, Art. no. 72.

[34] J. L. H. C. Reiss, and J. Wilkes, "Google–cluster traces: Format +schema," Menlo Park, CA, USA, Google Inc., White Paper, 2011.

[35] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency Comput., Prac. Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.
[36] I. Traffic, "Internet traffic archive," 1995. [Online]. Available: http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html
[37] The grid workloads archive, 2019. [Online]. Available: http://gwa.ewi.tudelft.nl
[38] Auvergrid, 2019. [Online]. Available: http://www.auvergrid.fr
[39] Nordugrid, 2019. [Online]. Available: http://www.nordugrid.org
[40] Sharcnet, 2019. [Online]. Available: https://www.sharcnet.ca
[41] P. Singh, P. Gupta, and K. Jyoti, "TASM: Technocrat ARIMA and SVR model for workload prediction of web applications in cloud," *Cluster Comput.*, vol. 22, no. 2, pp. 619–633, 2019.
[42] F. Farahnakian, P. Liljeberg, and J. Plosila, "LiRCUP: Linear regression based cpu usage prediction algorithm for live migration of virtual machines in data centers," in *Proc. 39th Euromicro Conf. Softw. Eng. Adv. Appl.*, 2013, pp. 357–364.
[43] F. J. Baldan, S. Ramirez-Gallego, C. Bergmeir, F. Herrera, and J. M. Benitez, "A forecasting methodology for workload forecasting in cloud systems," *IEEE Trans. Cloud Comput.*, vol. 6, no. 4, pp. 929–941, Oct.–Dec. 2016.
[44] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS," *IEEE Trans. Cloud Comput.*, vol. 3, no. 4, pp. 449–458, Oct.–Dec. 2015.
[45] Q. Zhang, L. T. Yang, Z. Yan, Z. Chen, and P. Li, "An efficient deep learning model to predict cloud workload for industry informatics," *IEEE Trans. Ind. Inform.*, vol. 14, no. 7, pp. 3170–3178, Jul. 2018.
[46] I. Rodríguez-Fdez, A. Canosa, M. Mucientes, and A. Bugarín, "STAC: A web platform for the comparison of algorithms using statistical tests," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2015, pp. 1–8.

Ashutosh Kumar Singh (Senior Member, IEEE) received the PhD degree in electronics engineering from the Indian Institute of Technology (BHU) Varanasi, India. He is currently a professor and the head of the Department of Computer Applications, National Institute of Technology Kurukshetra, India, and a charted engineer from U.K. He was a postdoc with the Department of Computer Science, University of Bristol, U.K. He has more than 20 years research and teaching experience in various universities of India, the U.K., and Malaysia. He has authored or coauthored more than 270 research papers in different journals, conferences, and news magazines. His research interests include verification, synthesis, design and testing of digital circuits, data science, cloud computing, machine learning, security, and big data.

Deepika Saxena received the MTech degree in 2014 in computer science and engineering from Kurukshetra University, Kurukshetra, Haryana, India, where she is currently working toward the PhD degree with the Department of Computer Applications, National Institute of Technology. Her research interests include neural networks, evolutionary algorithms, resource management, and security in cloud computing.

Jitendra Kumar received the doctorate degree from the National Institute of Technology, Kurukshetra, India, in 2019. He is currently an assistant professor with the Department of Computer Applications, National Institute of Technology, Tiruchirappalli, India. His current research interests include cloud computing, machine learning, data analytics, and parallel processing.

Vrinda Gupta received the BE degree in electronics and communication engineering from from Nagpur University in 1987, the MTech degree in electronics and communication engineering from Kurukshetra University in 1994, and the PhD degree in electronics and communication engineering from the National Institute of Technology (NIT) Kurukshetra, Kurukshetra, India, in 2017. She is currently an associate professor with Electronics and Communication Engineering Department, NIT Kurukshetra. Her research interests include computer communications, wireless communications and networking, wireless sensor networks, and the Internet of Things.