IEEE Second International Conference on Data Stream Mining & Processing
August 21-25, 2018, Lviv, Ukraine

Cloud Datacenter Workload Prediction Using Complex-Valued Neural Networks

Igor Aizenberg
Department of Computer Science
Manhattan College
New York, USA
igor.aizenberg@manhattan.edu

Kashifuddin Qazi
Department of Computer Science
Manhattan College
New York, USA
kqazi01@manhattan.edu

***Abstract*—Cloud computing infrastructures and datacenters depend on intelligent management of underlying CPU, memory, network, and storage resources. A variety of techniques such as load balancing, load consolidation, and remote memory allocation are used to maintain a fine balance between conflicting goals of high performance, and low costs and energy consumption. To meet these goals, successful prediction of the workloads is an important problem. By accurately predicting the resource utilization of host machines, datacenter owners can better manage the available resources. This paper presents a host resource usage prediction approach, based on a Multilayer Neural Network with Multi-Valued Neurons (MLMVN). An enhancement is further implemented to MLMVN to make it suitable for cloud datacenter applications. The approach is evaluated on real world load traces from Google's cluster data, as well as two grid based load traces. The algorithm is compared against some current state-of-the-art host-load prediction algorithms to show its accuracy, as well as performance gains.**

***Index Terms*—cloud datacenter, workload prediction, complex-valued neural networks**

I. Introduction

Cloud computing provides on-demand computing resources such as CPU, memory, networking, and storage for a variety of big data applications. The infrastructure for implementing clouds is a pool of resources hosted on large clusters of physical machines in datacenters. The sharing of this pool of resources within host machines provides benefits of scale, and results in low costs to use, high performance, energy savings, and elasticity. A variety of dynamic, online techniques such as load balancing, load consolidation, remote memory allocation, memory ballooning, etc. have been implemented to achieve the goals. However, using these techniques without considering the ever-changing resource requirements of the hosts can lead to severe issues. For example, consolidating hosts with large resource requirements can result in resource starvation, which in turn will cause degradation of performance. Similarly, remote memory may allow one host to access large amounts of memory, but may degrade performance on another host that is offering the remote memory. Therefore, it is imperative to accurately predict future host resource usage (host load) in advance, to intelligently make these management decisions. In general, longer (hours), but finer-grained predictions allow better management than shorter (minutes), coarse predictions.

Another concern while predicting cloud workloads, is the computational speed of the solution used. The time spent predicting is overhead performed in real time, and needs to be minimized, while maintaining a high accuracy.

Multi-layered neural networks with multi-valued neurons (MLMVN) are complex-valued neural networks with a derivative-free backpropagation learning algorithm. They have been shown to outperform competing machine learning techniques in a number of problems unrelated to cloud computing ([1], [2], [3]). MLMVNs converge quickly, and can predict multiple steps into the future without a large propagation of previous errors. Additionally, as shown by [4], complex-valued neural networks could be converted to algorithms suitable for quantum computing. The speed-up due to utilizing quantum computers in the prediction phase of datacenters will allow even finer grained prediction across thousands of machines. A preliminary version of this approach has been briefly discussed in [5]. All of this motivates the application of MLMVN to the problem of host load prediction for cloud datacenters.

The main contributions of this paper include a novel cloud host load prediction mechanism based on a modified MLMVN. Second, the paper describes an enhancement to MLMVN that mitigates some of the issues associated with it, making it suitable for host load prediction in a cloud computing environment. The overall approach is rigorously evaluated, and compared against a number of other host load prediction methods. Through all the results, the approach shows state-of-the-art short-term and long-term prediction performance, in terms of both accuracy and computational speed.

The rest of the paper is divided as follows. An overview of current host load prediction techniques is given in Sect. 2. The proposed method is described in Sect. 3. Experiment results and comparisons are presented in Sect. 4. The paper is concluded in Sect. 5.

II. Related Work

According to [6], research in host load prediction has been performed using Bayesian networks, artificial neural networks, decision trees, support vector machines, arima models, and cubic smoothing splines. Further, a number of other mathematical tools have been employed to solve the problem including

978-1-5386-2874-4/18/$31.00 2018 IEEE

Lviv Polytechnic National University Institutional Repository http://ena.lp.edu.ua

Fourier transforms [7], chaos theory [8], and fractal methods [9]. For grid-based systems, [10] utilized the Markov model for single-step predictions while [11] proposed a Kalman filter and autoregressive (AR) based multi-step method. The work in [12] used a feedforward artificial neural network (ANN) and achieved higher accuracy than other methods.

A comparison between Google cluster and grid system host loads shows that Google loads exhibit higher variance and noise, with lower seasonality [13]. Hence, it is more challenging to predict host loads in a cloud than in a grid. It has been shown that classical time series prediction methods such as arima models underperform when used in cloud environments [14].

The authors of [14] proposed a cloud load prediction method based on Bayes model. However, this approach only predicted the mean host load value, and did not aim to make fine-grained predictions. For fine-grained predictions, [15] proposed combining phase space reconstruction with an evolutionary algorithm. This method showed limited capability in multi-step ahead predictions [16]. Moreover, the prediction performance of this method is closely related to the parameters chosen, as the evolutionary algorithm is a stochastic global search method which may get stuck in local optimas [17].

A literature review reveals two methods that currently offer the best accuracy results for fine-grained host load prediction.

The first method used an autoencoder as the pre-recurrent feature layer of an echo state network (ESN) to perform host load predictions multiple steps in the future [17]. However, this method has some limitations. ESN uses a manually chosen leaking rate to control the degree of delays, which reduces its generalization ability when applied to different load traces. Moreover, the random initialization of a large reservoir, could degrade the performance of ESN when applied to noisy loads.

Currently, the approach described by [16] yields the best prediction accuracy for host load time-series prediction. They used the long short-term memory (LSTM) model in a recurrent neural network (RNN). By learning long-term dependencies, this approach demonstrated high accuracy for a large number of timesteps into the future even for noisy cloud host loads. However, RNNs suffer from computationally long training times due to the backpropagation algorithm being applied to recurrent layers. Even with optimizations, feed-forward neural networks are generally faster than RNNs.

As opposed to these approaches, the proposed prediction mechanism utilizes MLMVN which is a feed-forward neural network, utilizing complex-valued neurons. This offers higher functionality, better generalization capability and simplicity of learning. Additionally, MLMVN learning is derivative-free, and avoids falling into local optimas [2]. The method is used to achieve both short-term and long-term predictions.

III. Background

This section discusses the core details of MLMVN (input/output structure, neuron structure, activation function, error-correction rule, and backpropagation algorithm) to reproduce the results of this paper. Interested readers are encouraged to refer to [2] for a more thorough understanding of MLMVN.

A. Multi-Valued Neurons and MLMVN

The main distinction of MLMVN as compared to the classical feedforward neural network, is that its building blocks are Multi-Valued Neurons (MVN) with complex-valued weights. Using complex-valued inputs/outputs, weights and activation functions, it is possible to increase the functionality of a single neuron and a neural network, to improve their performance, and to reduce the training time ([3], [4]).

MVN was initially introduced as a discrete MVN in [2]. A continuous MVN was then introduced in [18]. This paper employs a continuous MVN. It implements a mapping between n inputs and a single output. All real-valued inputs (x_r) need to be in the range [0.0, 1.0] and are initially transformed to complex-valued inputs (x). The complex-valued outputs (y) are transformed back to real values (y_r) at the end.

$$x = e^{ix_r}$$

$$y_r = arctan2(y_{imag}, y_{real})$$

While MVNs inputs and output are complex numbers located on the unit circle, its weights are arbitrary complex numbers. An input/output mapping of a continuous MVN is described by a function of n variables

$$f(x_1, ..., x_n) = P(w_0 + w_1x_1 + ... + w_nx_n)$$

where $x_1, ..., x_n (x_j \in E_k, j = 1, ..., n)$ are neuron inputs and $w_0, w_1, ..., w_n$ are the weights. P is the activation function given by

$$P(z) = e^{iArg(z)} = z/|z|$$

where $z = w_0 + w_1x_1 + ... + w_nx_n$ is the weighted sum. $Arg(z)$ is the main value of the argument of the complex number z. Thus a continuous MVN output is a projection of its complex-valued weighted sum onto the unit circle.

The MVN learning for hidden neurons is based on the error-correction learning rule as described in [2], [18].

Using the described MVNs, the MLMVN is constructed. The MLMVN backpropagation learning algorithm is derivative-free and it is based on the generalization of the error-correction learning algorithm for a single MVN. This algorithm was proposed in [2] where it is described in detail. The batch version of this algorithm, used for this paper was proposed in [19].

B. Time Series Forecasting

Previously MLMVN has been considered for time series prediction of oil well production [1]. Time series forecasting can be formulated as a classification problem, with n past values of the time series (host loads in this case) as inputs and the $n+1$st value as the output. Presumably, there exists some functional dependence among the series members, according to which the $n+1$st member's value is a function of a certain number of preceding n members' values.

$$x_{n+1} = f(x_0, ..., x_n)$$

Lviv Polytechnic National University Institutional Repository http://ena.lp.edu.ua

TABLE I
PARAMETERS FOR MLMVN AND LSTM-RNN

Network Parameters	MLMVN	LSTM-RNN
No. of Inputs	24	24
No. of neurons in Hidden layer 1	32	128
No. of neurons in Hidden layer 2	64	-
Learning rate	1/(no.(weights))	0.05
Iterations	150	90

Suppose the historical data of host load usage is the time series $x_0, x_1, ..., x_r$. A training set is formed from this time series by using overlapping subsets of $n+1$ values. MLMVN is trained on the training set, using n values as input and the $n+1$st value as the desired output in each subset. The network weights learned through training can be thought of as an approximation of the function f. Therefore, using this approximate $\hat{f}$, future values of host loads can be predicted as follows:

$$\hat{x}_{r+1} = \hat{f}(x_{r-n+1}, ..., x_r)$$
$$\hat{x}_{r+2} = \hat{f}(x_{r-n+2}, ..., x_r, \hat{x}_{r+1})$$

where $\hat{x}$ is the predicted value of x. Multi-step ahead prediction is achieved by performing repeated one-step ahead prediction. At each step the predicted value is used as part of the next step's input. Thus, as many points as required in the future (called prediction window) can be predicted.

C. Minimizing Dependency on Initial Weights

A potential drawback of using MLMVN, is its high dependence on the initial random weights chosen. This results in the quality of network training varying heavily across different runs for the same dataset. To mitigate this problem, a batch learning algorithm based on complex QR decomposition was introduced for MLMVN in [20]. In [19], a linear least squares (LLS) based batch algorithm was proposed for a complex-valued neural network with a $single$ hidden layer. The algorithm proposed adding adjustment factors to all the weights after each iteration of training. The algorithm works as follows. In each iteration of training 1) Calculate the errors for all training samples using current weights. 2) Create an overdetermined system of linear algebraic equations for 'adjustment factors'. This system is given by the following

$$\Delta w_0^h + \Delta w_1^h x_1^j + ... + \Delta w_n^h x_n^j = \delta_j^h$$

Where, for the hth neuron, and the jth training sample, Δw_i^h is the weight adjustment factor for the neuron's ith weight, x_i^j is the ith input, and δ_j^h is the calculated error for the neuron. 3) Solve this system of equations for adjustment factors using LLS. 4) Adjust weights of all neurons in a layer simultaneously by adding the adjustment factors to the corresponding weights.

This resulted in faster learning, ability to maintain big learning sets, and improved generalization capability. For this paper, the algorithm is generalized and implemented for MLMVN with multiple hidden layers. For the rest of the paper, MLMVN refers to the modified MLMVN.

IV. EXPERIMENTAL EVALUATION

A. Experimental Setup

To evaluate the accuracy of the MLMVN based host load prediction algorithm, experiments were performed on two types of datasets. The cloud-based, noisy data from Google clusters, and the more regular grid data from sahara and themis datasets. The MLMVN model was built in Matlab. A network with two hidden layers was chosen based on observations from 100 host loads outside of the ones used for the experimental evaluations. The choice of two hidden layers is intuitive for time series predictions, since using only a single layer acts as a powerful low pass filter, and averages the output, ignoring local changes of the series to be predicted. The two layers consisted of 32 and 64 neurons respectively. Similar to previous methods, the number of inputs (preceding values considered for prediction) was chosen to be 24.

For all the datasets, MLMVN is compared against LSTM based RNN, and ESN. The two frameworks were built in Python using the TensorFlow library, to the specifications in [16] and [17]. The MLMVN neural network parameters are listed in Table I. For comparison, LSTM-RNN model's optimal parameters are also shown. To compare the accuracy of the various methods, the Mean Square Error is calculated as

$$MSE = 1/N \sum_{i=1}^{N} (y_i - p_i)^2$$

where N, y_i, and p_i are the length of the prediction window, actual values, and predicted values respectively.

B. Google Cluster Data

The Google cluster dataset [21] traces approximately 670000 jobs, and 40 million tasks across 12000 host machines during 29 days. The trace includes a variety of parameters, including CPU and memory usage of the tasks, as well as the location of the tasks on host machines. For this evaluation, the CPU usage of individual host machines for the 29 days were utilized to compare all methods. These values were obtained by aggregating the CPU usages of all the tasks residing on a host (located under the task_usage directory) during a time sample. The cluster data provides information every 5 minutes. Thus, the 29 day load trace for each host consists of 8352 data points. Since the Google trace provides CPU usage as a fraction of utilization in the range [0.0, 1.0], no further scaling is performed for these evaluations. For reference, the host load of one machine is shown in Fig. 1. The lack of any obvious pattern should be clear from this figure.

Similar to the other methods being compared, the first 26 days were used as training/ validation sets, while the last 3 days (day 27 to day 29) were used for testing. The results that follow indicate predictions for the testing set.

To demonstrate the efficacy of MLMVN based load prediction, predictions were performed on 4000 host machines from the Google cluster. These multi-step predictions were

Lviv Polytechnic National University Institutional Repository http://ena.lp.edu.ua

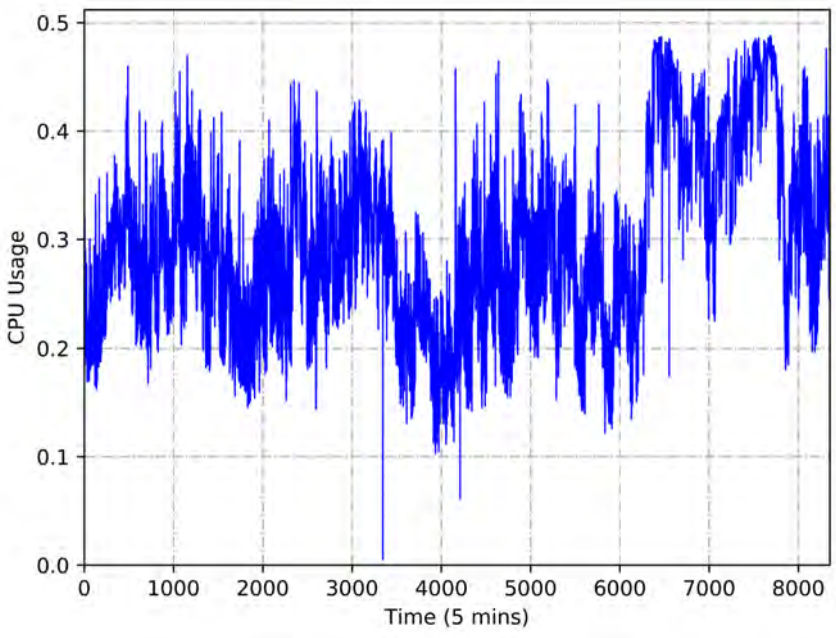


Fig. 1. Load trace for sample Google host - 29 days

done for prediction windows of 30 minutes (6 steps), 1 hour (12 steps), 1.5 hours (18 steps), 2 hours (24 steps), 2.5 hours (30 steps), and 3 hours (36 steps). Fig. 2 shows the MSE achieved by MLMVN, LSTM-RNN, and ESN for each of these prediction window sizes. It can be observed that MLMVN outperforms the other two prediction methods, and maintains its better accuracy even as the prediction window size is increased. Specifically for predictions 30 minutes into the future, MLMVN shows an MSE of 0.0032.

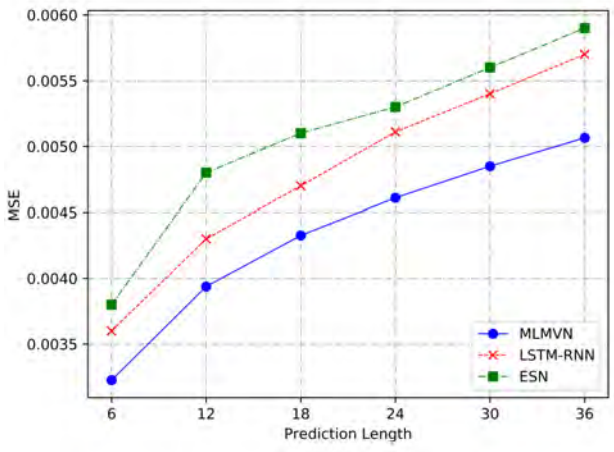


Fig. 2. Average MSE comparison for different prediction windows

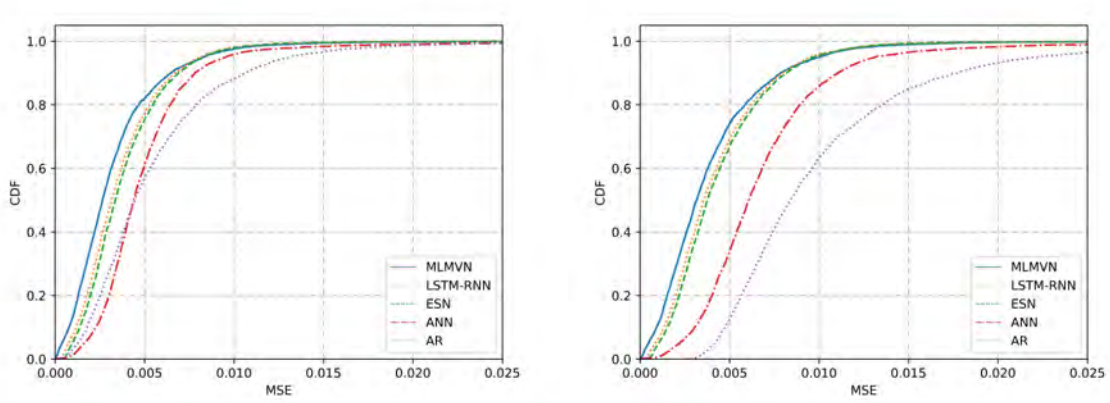
For additional comparisons on the Google cluster data, the Cumulative Distribution Functions (CDFs) of MSEs for two prediction windows (30 minutes and 1 hour) are presented in Fig. 3. For reference, results of real-valued feed-forward neural networks (ANN) and Hybrid-Autoregression (AR) built according to [12] and [11] are also shown. It can be deduced, that MLMVN outperforms all other methods, for all prediction window sizes. Methods such as the hybrid AR have a large variance in their accuracy, as opposed to MLMVN which shows consistent accuracy for most of the Google cluster hosts. For 30 min predictions, about 42% of predictions made by the hybrid AR show an MSE greater than 0.005, compared to approximately 18% for MLMVN. The benefits are even more apparent in the 60 min predictions.

C. Grid Data

In order to evaluate MLMVN based predictions on a different type of load trace, experiments were also performed on the grid-based loads provided by [22]. Specifically, two traces were chosen, namely the sahara and themis traces. These data traces include four days worth of data, sampled at one second each, for a total of 345600 data points each. The sahara dataset belongs to a compute server, while the themis dataset belongs to a desktop. The load is the number of processes that are running or are ready to run (the length of the scheduler's ready queue). For the original trace, the kernel sampled the length of the ready queue at a fixed rate, and averaged a window of previous samples to produce a load average.

For evaluation in this paper, the load traces were scaled to a range of [0.1, 0.9]. Similar to other methods being compared, each load trace was normalized using

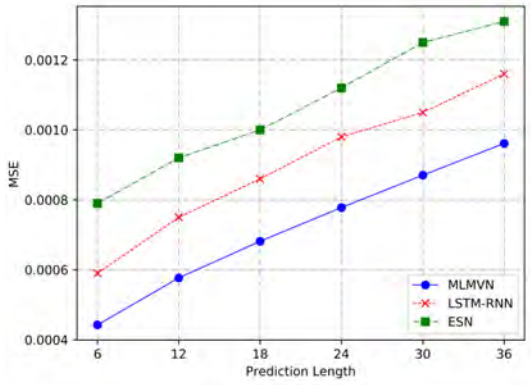
Lviv Polytechnic National University Institutional Repository http://ena.lp.edu.ua



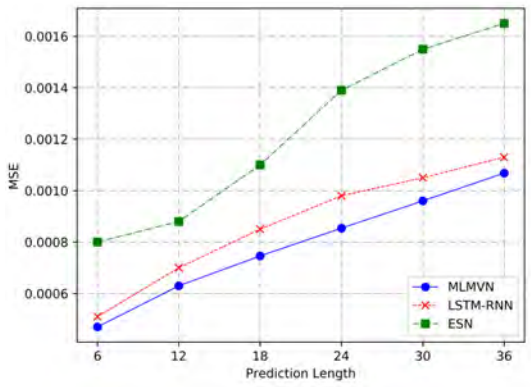
(a) 30 min prediction

(b) 60 min prediction

Fig. 3. CDFs of prediction MSEs for different methods



(a) sahara



(b) themis

Fig. 5. Average MSE of prediction in two grid load traces.

$$x_i = 0.1 + \frac{x_i - x_{min}}{x_{max} - x_{min}}(0.8)$$

where x_{max} and x_{min} are the maximum and minimum value of the load trace, respectively. For reference, a sample of the sahara trace is shown in Fig. 4. It can be observed, that compared to the Google cluster data trace, this trace is more regular, and has a visible pattern to it.

To compare accurately with the other state of the art methods, each load trace was split 80% of its length into a training set and the rest was the testing set. The prediction results are shown in Fig. 5. The results for LSTM-RNN and ESN are as reported by [16]. According to the results, MLMVN shows higher accuracy as compared to both LSTM-RNN and ESN, for all the prediction lengths.

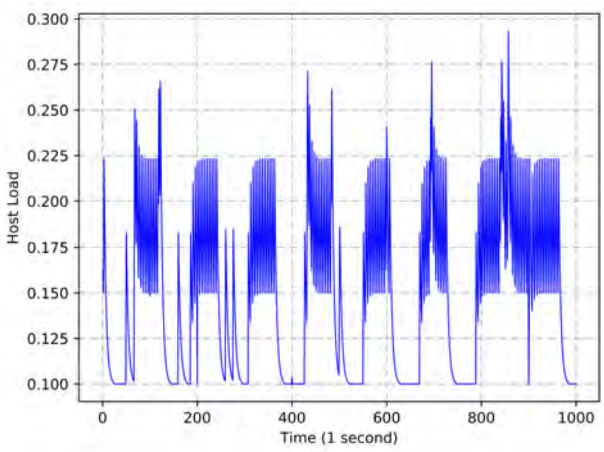


Fig. 4. Load trace sample from grid-based sahara data

The accuracy gains of MLMVN vs LSTM-RNN are least prominent in the themis dateset, compared to other datasets. The reason is that of all the loads evaluated, the themis dataset appears to be most regular, and predictable. Thus, most sophisticated algorithms achieve satisfactory performance. In spite of this, there is a visible improvement in accuracy with MLMVN across all prediction windows.

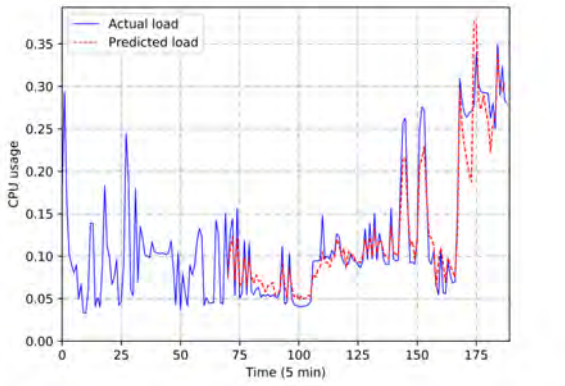
Lviv Polytechnic National University Institutional Repository http://ena.lp.edu.ua

TABLE II
TIME TO TRAIN VS ACCURACY

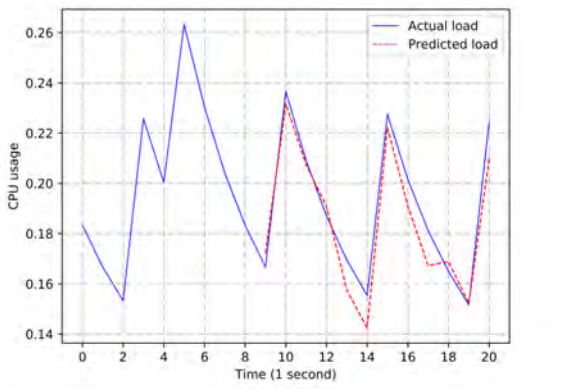
Method	Time to train (s)	Avg. 6 step MSE
MLMVN	38	0.0032
LSTM-RNN	105	0.0036
ANN	35	0.0052

D. Prediction Samples

In order to identify the difference in quality of MLMVN predictions between the cloud and grid loads, Fig. 6 presents 30 minute predictions in a sample from the Google cluster data, as well as the sahara data set. It can be observed that in the more regular sahara dataset, the predictions follow the actual loads closely. On the other hand, as expected, in the noisy Google cluster sample, the predictions follow the actual loads, albeit not as closely as the sahara sample.



(a) Google cluster sample



(b) sahara sample

Fig. 6. Actual loads vs Predicted loads

E. Time to Train Analysis

As previously noted, RNNs tend to train slower than similar Feed Forward Neural Networks. The implementations used for evaluation in this paper were executed on Ubuntu 16.04 machines with 8GB of RAM. The avg. training time for a single host in the Google cluster data for MLMVN was 38s versus LSTM-RNN's 105s. Table II showcases the results for 6 step predictions. MLMVN offers the most beneficial balance of time to train and accuracy. It should be noted that the MLMVN implementation is on Matlab, which should have a general disadvantage in computational speeds versus Python. Moreover, LSTM-RNN needs to train separately for each prediction window size needed (6 steps, 12 steps, etc). MLMVN uses iterative prediction, thus after training only once, as many future steps as required can be predicted.

F. MLMVN vs Vanilla-MLMVN

To compare the difference between the MLMVN (with LLS algorithm) used in these evaluations and vanilla-MLMVN (without LLS algorithm), the networks were trained on the same host load 10 times. The standard deviation of MSEs for MLMVN was 0.000067 as opposed to vanilla-MLMVN's 0.007. The best MSE in each case was 0.001701. Thus, while both methods achieve similar MSEs, the vanilla-MLMVN needs to be trained multiple times to obtain best results. In contrast, MLMVN trains consistently, making it suitable for online, real-time predictions in cloud computing environments.

V. CONCLUSION

This paper proposed an approach to predict host resource usage in cloud environments using complex-valued neural networks (MLMVN). The real-time prediction of host loads could be utilized for a variety of datacenter management concerns such as load balancing, load consolidation, remote memory allocation, etc. Through extensive experimental analysis, it was demonstrated that the proposed prediction solution produces state-of-the-art accuracy for real world Google cluster and grid load traces. Additionally, an analysis of the computational time revealed its superiority compared to other high-accuracy approaches that utilize recurrent neural networks. The complex-valued network also has the potential to be transformed into a quantum computing algorithm, which could offer even greater speed benefits in the future.

REFERENCES

[1] I. Aizenberg, L. Sheremetov, L. Villa-Vargas, and J. Martinez-Muñoz, "Multilayer neural network with multi-valued neurons in time series forecasting of oil production," *Neurocomputing*, vol. 175, pp. 980–989, 2016.

[2] I. Aizenberg and C. Moraga, "Multilayer feedforward neural network based on multi-valued neurons (MLMVN) and a backpropagation learning algorithm," *Soft Computing*, vol. 11, no. 2, pp. 169–183, 2007.

[3] N. N. Aizenberg and I. N. Aizenberg, "CNN based on multi-valued neuron as a model of associative memory for grey scale images," in *Cellular Neural Networks and their Applications, 1992. CNNA-92 Proceedings., Second International Workshop on*. IEEE, 1992, pp. 36–41.

[4] A. Hirose, *Complex-valued neural networks*. Springer Science & Business Media, 2012, vol. 400.

[5] K. Qazi and I. Aizenberg, "Towards quantum computing algorithms for datacenter workload predictions," in *Cloud Computing (CLOUD), 2018 IEEE International Conference on*. IEEE, 2018, p. In Press.

[6] N. Herbst, A. Amin, A. Andrzejak, L. Grunske, S. Kounev, O. J. Mengshoel, and P. Sundararajan, "Online workload forecasting," in *Self-Aware Computing Systems*. Springer, 2017, pp. 529–553.

Lviv Polytechnic National University Institutional Repository http://ena.lp.edu.ua

[7] Z. Gong, X. Gu, and J. Wilkes, "Press: Predictive elastic resource scaling for cloud systems," in *Network and Service Management (CNSM), 2010 International Conference on*. IEEE, 2010, pp. 9–16.

[8] K. Qazi, Y. Li, and A. Sohn, "Workload prediction of virtual machines for harnessing data center resources," in *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*. IEEE, 2014, pp. 522–529.

[9] M. Ghorbani, Y. Wang, Y. Xue, M. Pedram, and P. Bogdan, "Prediction and control of bursty cloud workloads: a fractal framework," in *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*. ACM, 2014, p. 12.

[10] S. Akioka and Y. Muraoka, "Extended forecast of cpu and network load on computational grid," in *Cluster Computing and the Grid, 2004. CCGrid 2004. IEEE International Symposium on*. IEEE, 2004, pp. 765–772.

[11] Y. Wu, Y. Yuan, G. Yang, and W. Zheng, "Load prediction using hybrid model for computational grid," in *Grid Computing, 2007 8th IEEE/ACM International Conference on*. IEEE, 2007, pp. 235–242.

[12] T. V. T. Duy, Y. Sato, and Y. Inoguchi, "Improving accuracy of host load predictions on computational grids by artificial neural networks," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 26, no. 4, pp. 275–290, 2011.

[13] S. Di, D. Kondo, and W. Cirne, "Characterization and comparison of cloud versus grid workloads," in *Cluster Computing (CLUSTER), 2012 IEEE International Conference on*. IEEE, 2012, pp. 230–238.

[14] ——, "Host load prediction in a google compute cloud with a bayesian model," in *High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for*. IEEE, 2012, pp. 1–11.

[15] Q. Yang, C. Peng, H. Zhao, Y. Yu, Y. Zhou, Z. Wang, and S. Du, "A new method based on psr and ea-gmdh for host load prediction in cloud computing system," *The Journal of Supercomputing*, vol. 68, no. 3, pp. 1402–1417, 2014.

[16] B. Song, Y. Yu, Y. Zhou, Z. Wang, and S. Du, "Host load prediction with long short-term memory in cloud computing," *The Journal of Supercomputing*, pp. 1–15, 2017.

[17] Q. Yang, Y. Zhou, Y. Yu, J. Yuan, X. Xing, and S. Du, "Multi-step-ahead host load prediction using autoencoder and echo state networks in cloud computing," *The Journal of Supercomputing*, vol. 71, no. 8, pp. 3037–3053, 2015.

[18] I. Aizenberg, C. Moraga, and D. Paliy, "A feedforward neural network based on multi-valued neurons," in *Computational Intelligence, Theory and Applications*. Springer, 2005, pp. 599–612.

[19] E. Aizenberg and I. Aizenberg, "Batch LLS-based learning algorithm for MLMVN with soft margins," in *Proceedings of the 2014 IEEE Symposium Series of Computational Intelligence (SSCI-2014)*. IEEE, 2014, pp. 48–55.

[20] I. Aizenberg, A. Luchetta, and S. Manetti, "A modified learning algorithm for the multilayer neural network with multi-valued neurons based on the complex QR decomposition," *Soft Computing*, vol. 16, no. 4, pp. 563–575, 2012.

[21] J. Wilkes, "More Google cluster data," Google research blog, Nov. 2011, posted at http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html.

[22] P. A. Dinda, "The statistical properties of host load," *Scientific Programming*, vol. 7, no. 3-4, pp. 211–229, 1999.

Lviv Polytechnic National University Institutional Repository http://ena.lp.edu.ua