PRACTISE: Robust Prediction of Data Center Time Series

Ji Xue*, Feng Yan*, Robert Birke†, Lydia Y. Chen†, Thomas Scherer†, and Evgenia Smirni*
*College of William and Mary
Williamsburg, VA, USA
{xuejimic,fyan,esmirni}@cs.wm.edu
†IBM Research Zurich Lab
Zurich, Switzerland
{bir,yic,tsc}@zurich.ibm.com

***Abstract*—We analyze workload traces from production data centers and focus on their VM usage patterns of CPU, memory, disk, and network bandwidth. Burstiness is a clear characteristic of many of these time series: there exist peak loads within clear periodic patterns but also within patterns that do not have clear periodicity. We present PRACTISE, a neural network based framework that can efficiently and accurately predict future loads, peak loads, and their timing. Extensive experimentation using traces from IBM data centers illustrates PRACTISE's superiority when compared to ARIMA and baseline neural network models, with average prediction errors that are significantly smaller. Its robustness is also illustrated with respect to the prediction window that can be short-term (i.e., hours) or long-term (i.e., a week).**

I. Introduction

Effective workload characterization and prediction hold the answers to the conundrum of efficient resource allocation in distributed and scaled out systems. Being able to *accurately* predict the upcoming workload within the next time frame (i.e., in the next 10 minutes, half hour, hour, or even week) allows the system to make proactive decisions, rather than reactive ones. Proactive decisions can be used with superior performance in storage systems by timely warming up the cache with the working set [1], [2], especially in systems where traditional internal work (e.g., garbage collection, snapshots, upgrades) is interleaved with the user workload during opportune times. Proactive scheduling of data analytics work can result in personalized advertising, sentiment analysis, or timely product recommendation, i.e., before the user leaves the site [3], [4], [5]. Virtual machine (VM) consolidation and migration is another example where accurate prediction of the physical machine utilizations can guide effective system usage [6], [7], [8]. In all of the above cases, prediction of the intensity of peak loads and of their timings becomes key to the effective launching of proactive management.

To maintain performance at tails, e.g., at high percentiles of response times, resource management policies [3], [6], [8], need to address the demands of peak loads instead of average loads only. Depending on the capability of predicting peak load magnitudes and timings, resources can be multiplexed at various degrees across users and across time. Such predictions can guide VM consolidation in data centers.

In this paper, we focus on data center workloads within the private cloud operated by IBM and used by major corporations for their IT needs. Prior work on workload characterization at IBM data centers [9] focused on statistical analysis of the usage of specific components of the virtual and physical machines, e.g., CPU and IO [9], [10]. This statistical analysis focused on averages, percentiles, and trends, aiming to a better understanding of how the workload evolves across a two-year period, but largely ignored the time series of the various performance metrics. In this paper, we focus on these time series and develop methodologies for accurate prediction of various workload metrics and especially peaks and their timings.

Classic time series models such as ARIMA [11] can be used for online prediction. Such models first need to be trained using past observations and can predict the upcoming workload. Alternatively, neural networks can be used in the same manner and provide a black box approach to predict the future, especially to predict events that have been observed in the past. Superior to the classic time series models that use a linear basis function, neural networks model input using non-linear functions, which improves their ability to handle more complex observations. Features gathered from observations are not all equally informative; some are relevant, while others are noise. Key to effective neural network prediction is the discovery of the appropriate features. Neural network training is then conducted based on these.

In this paper, we develop a robust framework for prediction of data center time series (PRACTISE) and illustrate the flexibility of such a black box approach by showing remarkable accuracy in usage prediction of data center workloads in the wild. We focus on four components: CPU, memory, disk, and network. We focus on an actual production workload and particularly on 56 physical machines that host 775 virtual machines during a time period of 61 days. Based on observations of the workload pattern and its periodicity, we extract the features that identify the time periods in which the repetitive patterns occur. We also develop a bagging module [12] and an online updating module to improve the stability, accuracy, and speed of PRACTISE. We provide detailed comparisons with ARIMA and show that the proposed black box approach offers a significant improvement in predicting resource usage, by reducing average errors by three times. PRACTISE slashes the false negative prediction rates of peak loads to less than 12% and achieves two fold to nine fold improvements in the accuracy of timing predictions. PRACTISE is lightweight and achieves training and prediction by an order of magnitude faster than other methods, which allows it to be used online.

This paper is organized as follows. Section II presents an overview of the workload. Section III presents the machine learning model. Section IV presents extensive experimental

978-3-901882-77-7

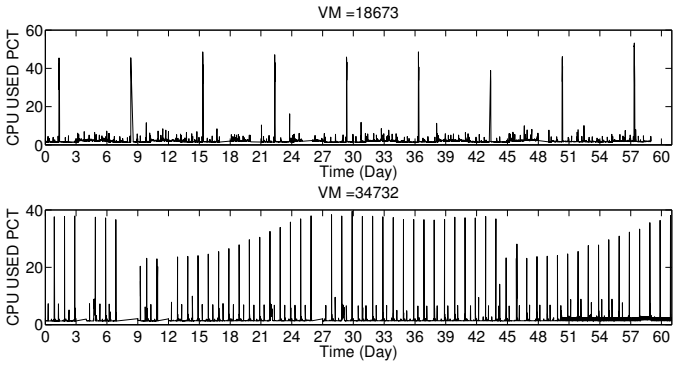

Fig. 1: CPU utilization over time for two different VMs.

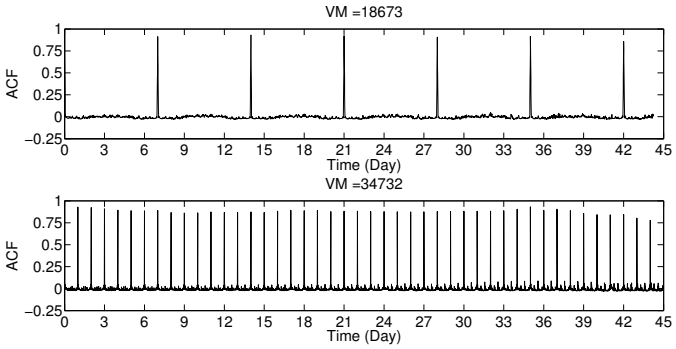

Fig. 2: Autocorrelation of CPU utilization for the two VMs of Figure 1.

evaluation. Section V discusses potential use scenarios. Section VI discusses related work. We conclude in Section VII.

II. VM Workloads in a Private Cloud

The target systems of this study are IBM private data centers, which are geographically distributed across all continents. These systems are used by various industries, including banking, pharmaceutical, IT, consulting, and retail, and are based on various UNIX-like operating systems, i.e., AIX, HP-UX, Linux, and Solaris. Those systems are highly virtualized, meaning that multiple virtual machines (VMs) are consolidated on a single physical box. Both VMs and boxes are very heterogenous in terms of resource configuration. The average virtualization level per box is ten [9]. We have collected resource utilization statistics from several thousands of VMs and boxes since February 2013. The finest observation granularity is 15 minutes[1]. The analysis here is based on two-month data from March 1, 2013 to April 30, 2013.

We focus on usage of four types of resources: CPU, memory, disk, and network. Using the base observation window of 15 minutes, we collect the following statistics:

- **CPU utilization**: the percentage of time the CPU is active over the observation window.
- **Memory utilization**: the percentage of memory capacity used.
- **Disk space usage**: the percentage of allocated disk space used.
- **Network bandwidth usage**: the total network traffic rate measured in mega bits per second (Mbps).

The collected trace data is retrieved via `vmstat`, `iostat`, and supervisor specific monitoring tools.

The VM workloads within the IBM private cloud exhibit clear periodic patterns over time [9], see Figure 1. The figure focuses on two different VMs and illustrates the CPU utilization within successive time windows of 15-minute across all 61 days. The upper plot shows a regular periodic pattern with a period of 7 days, while the bottom one shows a more complex pattern with clear trend changes. We also observe that similar periodic patterns exist in different resources, i.e., memory, disk, and network. To the interest of space, we do not present these results here. Pattern periodicity suggests that there exist opportunities for workload prediction.

To capture and quantify such periodic patterns, we perform statistical analysis of the workloads by computing the autocorrelation of the time series of CPU utilization. Autocorrelation is a mathematical representation of the degree of similarity in a time series and a lagged version of itself. As such, it is ideal for discovering repeating patterns by quantifying the relationship between different points of a time series as a function of the time lag [13]. The autocorrelation metric is in the range of $[-1, 1]$. Higher positive values indicate that the two points between the computed lag distance are "similar", i.e., have stronger correlation. Zero values suggest no periodicity. Negative values show that the two points lag elements apart are diametrically different. We show the autocorrelation of CPU resource usage for the two selected VMs in Figure 2. It is clear that the autocorrelation becomes high at certain lag values and that the lag values[2] can be different for different VMs. In the following section, we demonstrate how to utilize autocorrelation to select the appropriate features in order to train a neural network that can model the workloads accurately.

III. Methodology

A time series prediction model uses past observations to forecast future values. There are different ways to build the time series prediction model. Traditional time series e.g., the ARMA/ARIMA [11] and Holt-Winters exponential smoothing [14] are based on a linear basis function, and as a result they are not effective in predicting complex behaviors. In addition, these models are backward looking only methods, which makes it difficult to capture any new patterns that have not appeared before. Furthermore, the underlying approximation function usually lacks intuitive explanations. For all of the above reasons, it is difficult to improve the prediction accuracy of such types of models. On the other hand, neural networks are capable of modeling input as non-linear functions, which offers great potential in handling complex time series [15]. We start from the standard universal neural network toolbox provided by MATLAB [16] and then introduce PRACTISE by selecting more appropriate features, using bagging and online

[1]Collection of data is done by another IBM branch, therefore, we do not have any control on obtaining data at lower granularity.

[2]Note that a lag of 1 corresponds to two intervals 15 minutes apart.

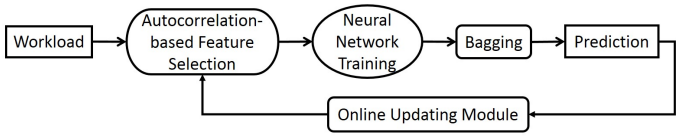

Fig. 3: Overview of PRACTISE.

updating to improve its accuracy and stability. An overview of PRACTISE is shown in Figure 3. The workload is fed to the autocorrelation-based feature selection module. The selected features then become inputs to the neural network training component. The bagging module processes the aggregated results. Finally, the online updating model monitors the prediction error and triggers a retraining if large errors are detected. In the following, we introduce each component in detail.

A. Universal Neural Network

Artificial neural networks are inspired by biological neural networks [17] and are composed of many interconnected *neurons*. The weights associated with the neurons are used to approximate non-linear functions of the inputs and are tuned during a training process. Discovering appropriate features is the key to building an accurate neural network model. The universal neural network toolbox provided by MATLAB uses a generalized algorithm for feature selection. To train a neural network, the input data set is usually divided into three subsets [18]: training, validation, and test. The neural network uses the training set to tune its weights and utilizes the validation set to determine the convergence point and prevent overfitting. The test set is used for evaluation of the training accuracy.

To understand the prediction accuracy of the standard neural network toolbox provided by MATLAB, we conducted extensive experiments. Figure 4 illustrates the default MATLAB prediction (tagged BaselineNN) of the utilization of the two VMs shown in Figure 1. We have trained and validated the neural network using the first 14 days, and we show here the results for days 15 to 24. The figure clearly shows the neural network's pitfalls as prediction accuracy is often poor. Using the standard MATLAB toolbox, the underlying feature selection algorithm is not tuned to optimize the information provided by the repeating patterns. Therefore, we are motivated to explore a better feature selection algorithm for selecting the appropriate features for the data center workloads that we have in hand.

B. Autocorrelation-based Features

Intuitively, appropriate features should reliably capture periodic behavior, changing trends, and repeating patterns. To identify the appropriate features, we resort to the correlogram in Figure 2 because autocorrelation can provide quantitative and qualitative information on the above factors. Figure 2 shows that there can be several lags with high positive autocorrelation values. This indicates that there exist several good candidate features that represent short-term to long-term correlation patterns. To automate the process, we use a local maximum detection function to identify the peak points in autocorrelations and use the respective lag values as features for neural network training. In this way, different correlation

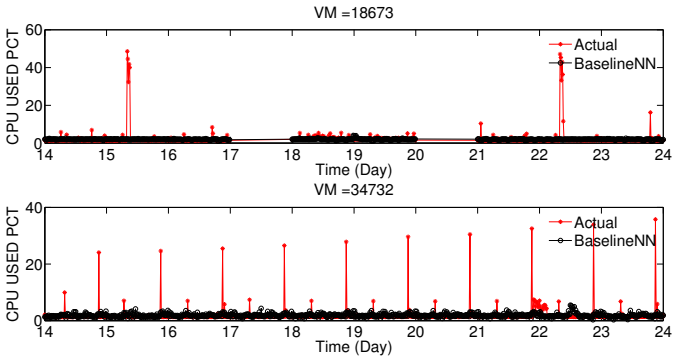

Fig. 4: CPU workload prediction by the neural network toolbox provided by MATLAB for two different VMs. The two gaps in the first plot are due to the VM being switched off.

TABLE I: Training time using 14 days' data and prediction length of 1 day.

VM ID	Training Time (sec)		Prediction Time (sec)	
	BaselineNN	PRACTISE	BaselineNN	PRACTISE
18673	300	30	257	10
34732	480	50	360	15

ranges from short-term to long-term can all be captured, which improves the effectiveness of the neural network. The remaining steps are the same as with the universal neural network toolbox provided by MATLAB. We stress that the feature selection process is fully automatic.

C. Bagging

The training features are not the sole factor in the prediction accuracy of a neural network model; the quality of the trained model also depends on other factors. The training data sets are another crucial factor [12]. As discussed earlier, the training set is split into training, validation, and test subsets. Different ways of splitting may result in different samples being used at different stages and therefore result in different trained models. In order to minimize the artificial effects caused by a certain splitting rule, we split the data set randomly several times (e.g., 20 times), and each split trains a different model. In other words, we train a group of neutral network models by using the same data set but with different splits. Each model has its own prediction result. The prediction results from different models together become a distribution of prediction results. To compute the final prediction results from the distribution of prediction results, we first use the 3-sigma rule [19] (e.g., 99% confidence interval) and z-score [20] (e.g., within [-0.85,0.85]) to filter out outliers and then compute the average of the remaining data as the final prediction. Bagging may not always guarantee that optimal prediction is achieved, but it consistently improves prediction accuracy compared to using only a single trained model.

D. Online Updating Module

In a real cloud environment, there can be sudden or permanent workload changes caused by unexpected events. As neural network models rely on past information to forecast the future, workload characterization changes may not be

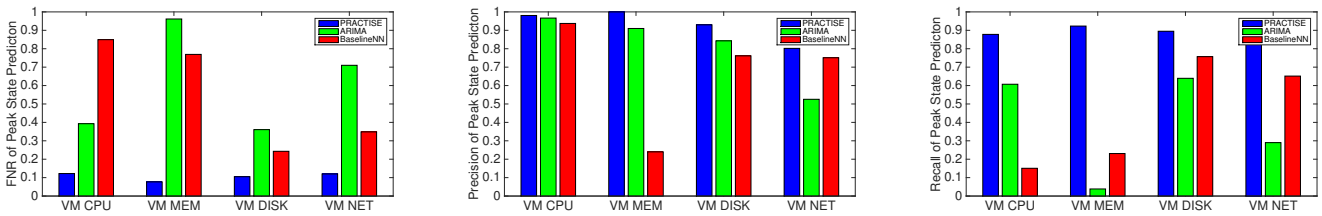

Fig. 5: Prediction accuracy for peak states. The left plot is the false negative rate (FNR) of peak state prediction, the middle plot is the precision of peak state prediction, and the right plot is the recall of the peak state prediction.

timely reflected in the prediction results. To ensure an agile response to workload characterization changes, we add an online updating module. The update is triggered based on monitoring the prediction errors periodically. If errors suddenly surge, a workload change is suspected and the neural network model is retrained. We emphasize that the computational cost of the neural network training and prediction is not significant thanks to the simple yet efficient feature selection process. Thus, it allows us to retrain the model online quickly at low cost. We demonstrate two examples in Table I to show how long PRACTISE takes for training and prediction compared to BaselineNN on a machine with 2.8 GHz Intel Core i7 CPU, 16 GB memory and 750 GB SSD. From the table, it is clear that both the training time and prediction time of PRACTISE are very low and that PRACTISE is one order of magnitude faster than BaselineNN. This difference may appear modest, but if prediction has to be done simultaneously for thousands of VMs and multiple resources, it becomes significant. The training and prediction times change linearly with the amount of training data and prediction length.

IV. Experimental Evaluation

We describe the methods used for the evaluation below:

- **ARIMA**: the standard ARIMA algorithm, used as baseline comparison.
- **BaselineNN**: the default setting of the neural networking toolbox provided by MATLAB, used as a second baseline comparison.
- **PRACTISE**: the workload prediction framework proposed in this paper.

Evaluation strategies. We use the first 14 days as training data and use the following 46 days as the data for evaluation of the prediction accuracy. With the online updating module, PRACTISE automatically triggers a retraining if the monitored error is outside the confidence interval determined by the 3 sigma rule [19]. We present the results for prediction length of 1 day ahead. We also present two more cases, one predicting 2 hours ahead (short window) and one predicting 1 week ahead (long window). We evaluate PRACTISE by comparing it to ARIMA and BaselineNN in two prediction scenarios: state predictions and timing, and quantified predictions.

State predictions and timing. Several scheduling and management frameworks [1], [21] do not require quantified prediction. Instead, workloads are classified into states, e.g., peak states with relatively high resource usage and non-peak states with relatively low resource usage[3], and only qualitative predictions are needed. In such a scenario, the quality of the prediction is measured by whether the future state can be predicted correctly. In addition, it is critical to be able to not only predict a peak state, but also the time *when* this state occurs. We quantify the timing of the predictions across all VMs in a cumulative way, i.e., we count for how many 15-minute intervals the timing of the prediction of the peak state is delayed.

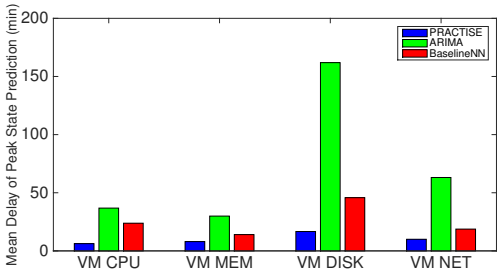

Fig. 6: Mean delay between prediction and actual occurrence of peak states.

With the given granularity of 15 minutes, every entry in the traces represents a peak or non-peak state. The threshold between peak and non-peak states is determined via K-means clustering. Because timely predicting peak states is utterly important, we first evaluate the accuracy of peak state prediction. The left plot in Figure 5 illustrates the rate of false negative peak state predictions, which is defined as the number of wrong peak predictions (i.e., states that are predicted as non-peak) divided by the total number of actual peak predictions. Results are across all VMs. PRACTISE consistently achieves less than 12% false negatives across all resources. The false negative rates of ARIMA and BaselineNN are much higher and very random across resources. We also provide two other commonly used metrics for evaluating prediction accuracy: precision, which is defined as the fraction of retrieved instances that are relevant and recall, which is defined as the fraction of relevant instances that are retrieved, see the middle and right plots in Figure 5 respectively. PRACTISE again consistently outperforms ARIMA and BaselineNN with respect to the two metrics, which further validates the accuracy of PRACTISE.

Figure 6 illustrates the average delay (in minutes) between the prediction and the actual occurrence of peak states across

[3]Here we focus on peaks because of literature [22], [23], but it can be applied to other utilization levels, not only peaks.

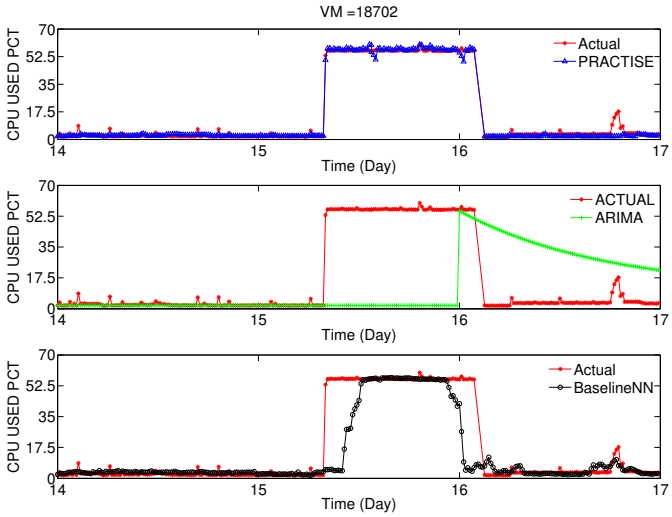

Fig. 7: Prediction for VM CPU utilization.

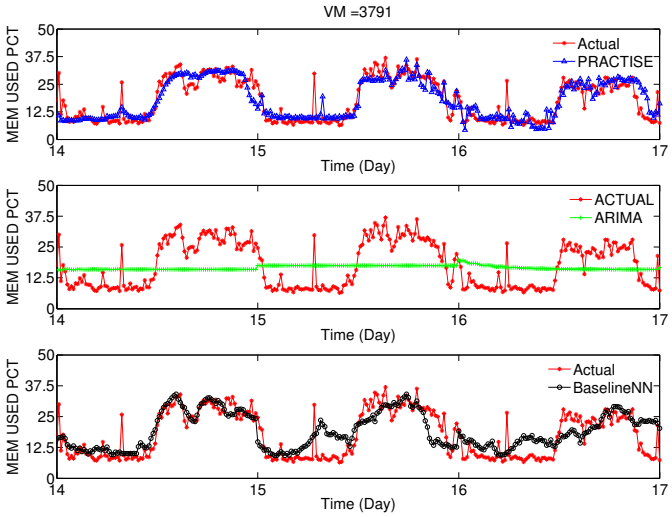

Fig. 8: Prediction for VM memory utilization.

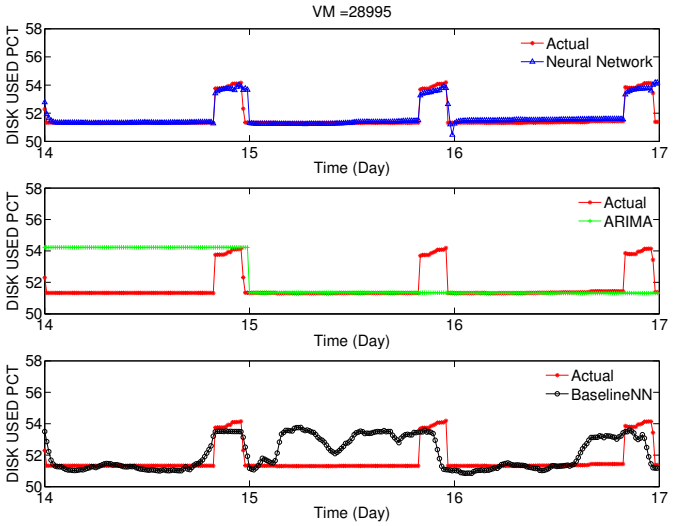

Fig. 9: Prediction for VM disk space usage.

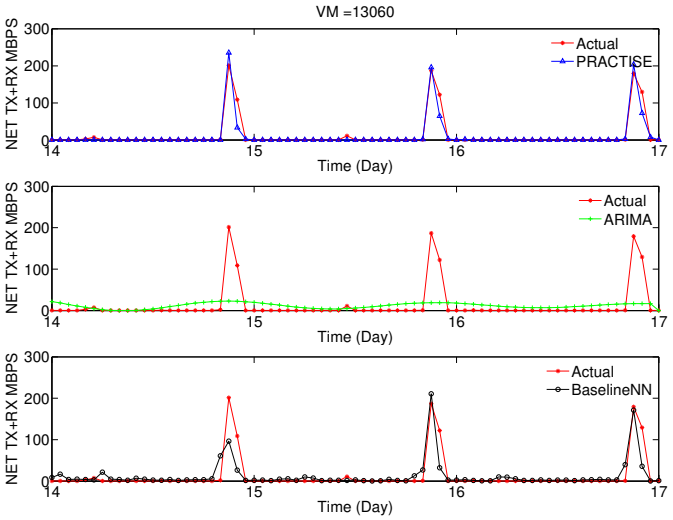

Fig. 10: Prediction for VM network bandwidth usage.

all 775 VMs. PRACTISE shows a remarkable accuracy across all resources with values dramatically outperforming all other methods, i.e., for VM CPU utilization, the average delay reduces from 36.80 minutes (ARIMA) and 23.82 minutes (BaselineNN) to 6.22 minutes; for VM memory utilization, from 29.93 minutes (ARIMA) and 14.00 minutes (BaselineNN) to 8.00 minutes; for VM disk space usage, from 161.88 minutes (ARIMA) and 45.75 minutes (BaselineNN) to 17.02 minutes; and, for VM network bandwidth usage, from 63.12 minutes (ARIMA) and 18.75 minutes (BaselineNN) to 10.05 minutes.

Quantified predictions. Scheduling and management frameworks do require quantified prediction [24], [25], especially for systems that need to meet certain service level objectives. We first show overtime plots for the actual workload and predicted results. Results for VM CPU utilization, VM memory utilization, VM disk space usage, and VM network bandwidth usage are shown in Figure 7, Figure 8, Figure 9, and Figure 10 respectively[4].

[4]We zoom in a 3-day period for clearer presentation.

Each point on the graphs corresponds to the finest workload granularity that is available, i.e., 15 minutes. It is clear that the prediction error of PRACTISE is consistently lower than both ARIMA and BaselineNN, especially for sudden workload surges, thanks to the more appropriate feature selection and bagging used in PRACTISE. Note that predicting sudden workload surges is very important as it can drive scheduling/management to allocate timely the right amount of resources to prevent performance pitfalls.

While the results presented before show cases of consistent periodicity across time, in Figure 11 we show a more challenging case where the trends of the periodical pattern change. The results show that PRACTISE can effectively capture this thanks to the online updating component. ARIMA can also react to the trend change, but it fails to capture most peak states. However, BaselineNN can only predict events that have been observed before, and are therefore unable to capture such trend changes. The experiments cover a variety of data center configurations and applications with various usage patterns, but due to the interest of space, we skip other results here.

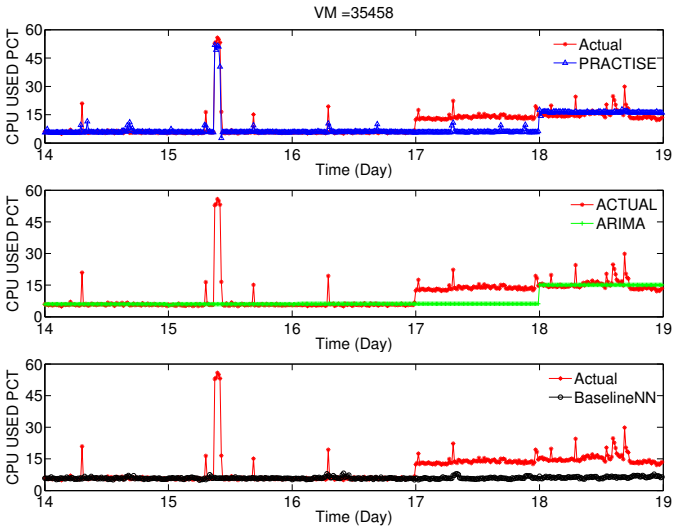

Fig. 11: Prediction for CPU utilization; the trends changes after day 17.

To quantify the prediction errors, we define the prediction error (PCT) as $\frac{|prediction-actual|}{actual}$. We show the CDFs for one specific VM (first column of Figure 12) for CPU, memory, disk, and network. We also show the mean and 90th percentile in the legend. The results illustrate clearly that PRACTISE is consistently superior in accuracy compared to ARIMA and BaselineNN as PRACTISE achieves up to 3 times better prediction accuracy than in terms of average prediction errors. The second column of Figure 12 illustrates the same information but cumulative across *all* 775 VMs. The superiority of PRACTISE is also clear in this comparison.

To demonstrate the importance of the bagging and the online updating modules, we also compare the prediction errors when bagging and online updating are activated, see Figure 13. The results indicate that with these two components, the prediction accuracy can be significantly further improved, which verifies that these two enhancements are non-trivial and useful.

Finally, we evaluate PRACTISE for different prediction lengths. We demonstrate the prediction length of 2 hours and 1 week in Figure 14. The results clearly illustrate that PRACTISE consistently outperforms ARIMA and BaselineNN for different prediction lengths. In addition, the change in the size of the prediction window does not affect robustness.

Challenging cases. PRACTISE relies on the autocorrelation values as features for neural network training. Here we evaluate a challenging case with poor autocorrelation structure, see the top plot of Figure 15. The figure illustrates the autocorrelation plot of the memory utilization of a VM. The autocorrelation function switches between positive and negative values, which makes feature selection very challenging. The CDF of prediction errors for this VM is presented in the bottom plot of Figure 15. The results suggest that even for this case, PRACTISE still achieves relatively good prediction accuracy and clearly outperforms ARIMA and BaselineNN. The robustness in prediction is due to the fact that PRACTISE does not solely rely on the autocorrelation features but also on bagging and online updating which contribute to more sophisticated predictions. We conclude that PRACTISE is effective, efficient, and robust.

V. Discussion

In this paper, we provided the first important step that is required for VM consolidation and/or load balancing: a framework for efficient and accurate prediction of future load, and in particular peak loads and their timing. This prediction is robust: even for cases where the workload burstiness does not have a clear repetitive pattern, we still manage to achieve remarkable accuracy and outperform ARIMA and generic neural network models for future time windows that can range from 1 hour to a week. There are many important implications of this work:

Dynamic VM consolidation driven by resource demands of different percentiles: PRACTISE can provide resource usage predictions for VMs but also for physical machines (PMs) where the various VMs may be consolidated. PRACTISE can provide different types of usage statistics, e.g., means, and percentiles. Indeed, for PMs, the traces provide the number of VMs per PM, as well as resource usage information. PRACTISE can be used to predict this information and drive different consolidation strategies[5] that are based on different load statistics. Due to the remarkable accuracy of PRACTISE on capturing the peak loads and their timings, the consolidation policy can aggressively conserve resources without risking performance degradation.

VM consolidation driven by prediction of multiple resources: PRACTISE is able to explore the availability of both CPU and memory on PMs. Most importantly, having predictions on multiple resources, one can focus on the most scarce resource and drive the consolidation policies accordingly. Even more specifically, since there is significant burstiness in both memory and CPU usage, a strong prediction model can really help in optimizing usage for both resources.

Minimizing the impact of VM migration: the VM migration overhead is known to be non-negligible and application performance can thus drastically degrade, especially when the migration timing collides with peak loads of the VMs or the underlying physical hosts. Intelligent migration can greatly leverage the information of future loads provided by PRACTISE and select the optimal timings for migrating VMs.

A bird's eye view of data center resource usage: beyond VM consolidation, accurate information on future loads of different resources enables a holistic load management of the entire data center, including IT, cooling, and energy costs. The server loads consume the energy to power up not only server resources but also the cooling facility. Fundamental questions, such as turning on/off components, can not be addressed without a holistic view of data center resource usage.

Other workloads: PRACTISE an be applied to any workloads with autocorrelation, e.g., storage workloads [1].

[5]PRACTISE shows excellent prediction results for the PMs in this data set. These results are not shown due to lack of space.

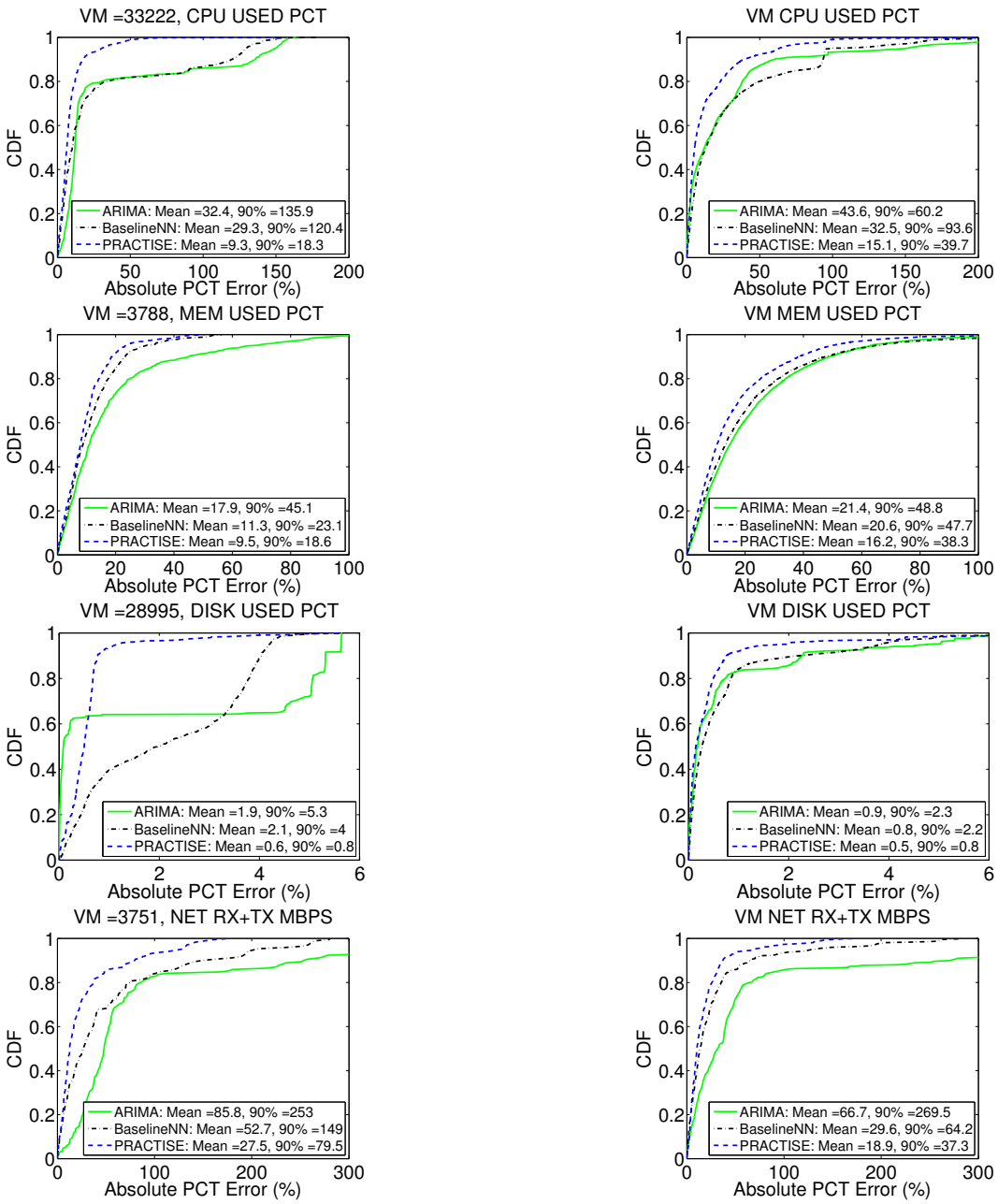

Fig. 12: Prediction error comparison for different prediction methods. The graphs are for VM CPU utilization (row 1), VM memory utilization (row 2), VM disk space usage (row 3), VM network bandwidth prediction (row 4). The first column is for a selected VM and the second column shows accumulated results over all VMs. Prediction length is 1 day ahead.

VI. Related Work

ARMA/ARIMA [11] have been widely used for time series prediction in several systems areas. Tran and Reed use ARIMA to improve block prefetching for scientific applications [26]. They use ARIMA to predict the temporal access pattern and Markov models to identify spatial access patterns and manage to identify what and how much to prefetch. Their predictor is implemented on the Linux file system. In [5] ARIMA is used for effective user traffic prediction for capacity planning. The authors focus on cost-efficient database replication that is driven by the anticipated user traffic within the LinkedIn social network. ARIMA models have also been used in sensor networks to reduce the frequency of sampling and to improve on energy efficiency by transmitting only deviations from the ARIMA-predicted values [27]. Anomaly detection is yet another area where ARIMA models have been used [28].

Machine learning techniques are used to overcome the limitation of the linear basis function of ARIMA models and are used for effective characterization of TCP/IP [29] and web server views [30]. Neural networks are used for performance prediction of the total order broadcast, which is a key building block for fault-tolerant replicated systems [31]. Ensembles of

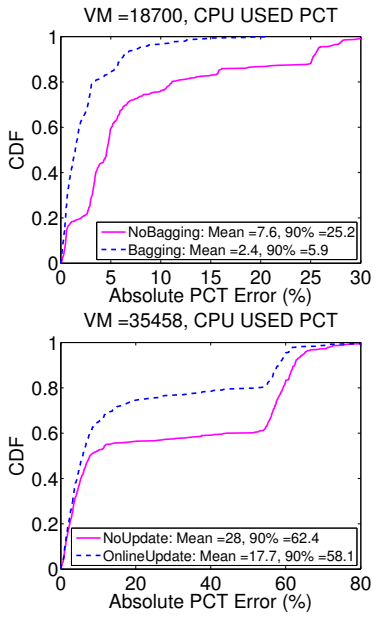

Fig. 13: Prediction error comparison of VM CPU utilization with and without bagging (top plot), and with and without online updating module (bottom plot).

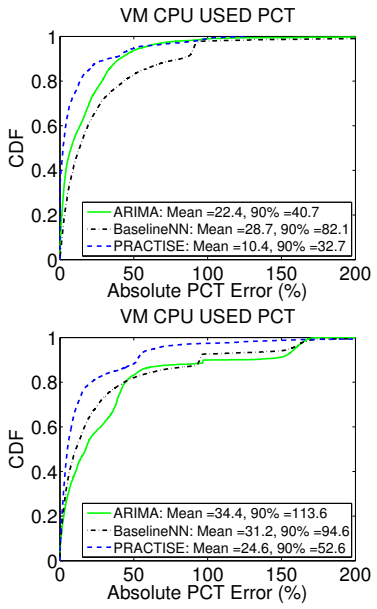

Fig. 14: Prediction error comparison of VM CPU utilization for prediction length of 2 hours (top plot) and 1 week (bottom plot). The results are accumulated across all VMs.

time neural network models have been used to project disk utilization trends in a cloud setting [32]. Neural networks and hidden Markov models are used for automatic IO pattern classification and are evaluated with both sequential and parallel benchmarks [21]. Probabilistic models that define workload states via Markov Modulated Poisson Processes have been used in [1] to interleave workloads with different performance objectives. Machine learning techniques have been widely used for workload prediction [33], [34], [35], [36]. In contrast to these works, we rely on the autocorrelation and automate the

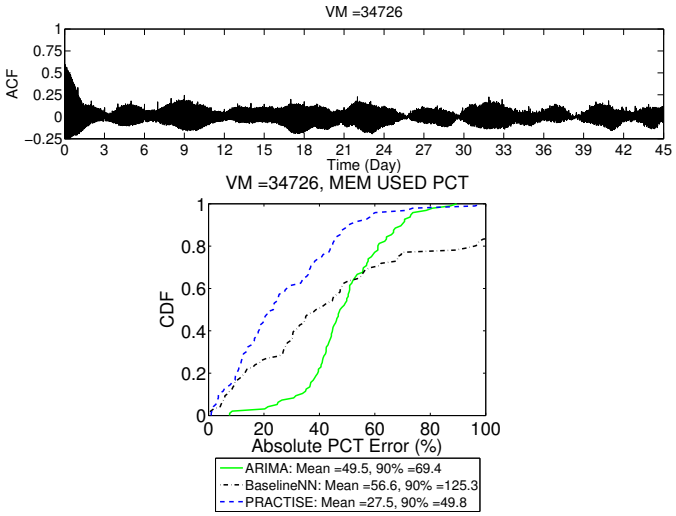

Fig. 15: A challenging case (VM 34726). Autocorrelation (top plot) and prediction error comparison (bottom plot) of memory utilization for different prediction methods.

entire learning process.

The effectiveness of the proposed neural network approach that we advocate in this paper is based on statistical analysis of the workload so that the most relevant features are selected for the training data set. Training the model with careful feature selection significantly improves its accuracy and stability but also increases the speed of training and prediction, making it appropriate to use for online performance prediction and capacity planning. In addition, due to the appropriate feature selection, PRACTISE can provide short-term (e.g., 15 minutes) to long-term (e.g., one day or one week ahead) predictions and achieve excellent accuracy. These superior predictions facilitate robust long-term capacity planning and resource allocation.

VII. Conclusion and Future Work

In this paper, we develop PRACTISE, an enhanced neural network based framework for predicting the usage of various resources in data centers. PRACTISE uses autocorrelation-based feature selection, boostrap aggregation, and online updating. We extensively evaluate PRACTISE on predicting CPU, memory, disk and network usage on a set of 775 VMs over a period of 2 months and compare its prediction effectiveness to ARIMA and basic neural network models. We are able to achieve up to 3 times better prediction accuracy in terms of average prediction errors and dramatic improvements (2- to 9-fold) with respect to the prediction timings. Thanks to the excellent prediction accuracy of PRACTISE, we are able to efficiently capture the peak loads in terms of their intensities and timing, in contrast to classic time series models. In our future work we intend to use PRACTISE to explore VM consolidation and migration policies tailored to cater to peak demands in a cost-effective way.

VIII. Acknowledgments

This work is supported by NSF grant CCF-1218758 and EU commission FP7 GENiC project (Grant Agreement No 608826).

REFERENCES

[1] J. Xue, F. Yan, A. Riska, and E. Smirni, "Storage workload isolation via tier warming: How models can help," in *Proceedings of the 11th ICAC*, 2014, pp. 1–11.

[2] Y. Zhang, G. Soundararajan, M. W. Storer, L. N. Bairavasundaram, S. Subbiah, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Warming up storage-level caches with Bonfire," in *FAST*, 2013, pp. 59–72.

[3] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin, and S. Babu, "Starfish: a self-tuning system for big data analytics." in *CIDR*, vol. 11, 2011, pp. 261–272.

[4] J. Cohen, B. Dolan, M. Dunlap, J. M. Hellerstein, and C. Welton, "Mad skills: new analysis practices for big data," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1481–1492, 2009.

[5] Z. Zhuang, H. Ramachandra, C. Tran, S. Subramaniam, C. Botev, C. Xiong, and B. Sridharan, "Capacity planning and headroom analysis for taming database replication latency: experiences with linkedin internet traffic," in *Proceedings of the 6th ACM/SPEC ICPE*, 2015, pp. 39–50.

[6] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *NSDI*. USENIX Association, 2005, pp. 273–286.

[7] M. Nelson, B.-H. Lim, G. Hutchins *et al.*, "Fast transparent migration for virtual machines." in *USENIX ATC*, 2005, pp. 391–394.

[8] T. Wood, P. J. Shenoy, A. Venkataramani, and M. S. Yousif, "Black-box and gray-box strategies for virtual machine migration." in *NSDI*, vol. 7, 2007, pp. 17–17.

[9] R. Birke, A. Podzimek, L. Y. Chen, and E. Smirni, "State-of-the-practice in data center virtualization: Toward a better understanding of VM usage," in *DSN*, 2013, pp. 1–12.

[10] R. Birke, M. Björkqvist, L. Y. Chen, E. Smirni, and T. Engbersen, "(big)data in a virtualized world: volume, velocity, and variety in cloud datacenters," in *FAST*, 2014, pp. 177–189.

[11] B. George, *Time Series Analysis: Forecasting & Control, 3rd ed*. Pearson Education India, 1994.

[12] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[13] L. M. Leemis and S. K. Park, *Discrete-event simulation: a first course*. Pearson Prentice Hall Upper Saddle River, NJ, 2006.

[14] P. Goodwin, "The holt-winters approach to exponential smoothing: 50 years old and going strong," *Foresight*, pp. 30–34, 2010.

[15] R. J. Frank, N. Davey, and S. P. Hunt, "Time series prediction and neural networks," *Journal of Intelligent and Robotic Systems*, vol. 31, no. 1-3, pp. 91–103, 2001.

[16] H. Demuth, M. Beale, and M. Hagan, "Neural network toolbox^{TM} 6," *User's Guide*, 2008.

[17] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*, 1st ed. Cambridge, MA, USA: MIT Press, 1995.

[18] T. Hill, M. O'Connor, and W. Remus, "Neural network models for time series forecasts," *Management Science*, vol. 42, no. 7, pp. 1082–1092, 1996.

[19] G. Upton and I. Cook, *A Dictionary of Statistics 3e*. Oxford university press, 2014.

[20] M. L. Marx and R. J. Larsen, *Introduction to mathematical statistics and its applications*. Pearson/Prentice Hall, 2006.

[21] T. M. Madhyastha and D. A. Reed, "Learning to classify parallel input/output access patterns," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 8, pp. 802–813, 2002. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TPDS.2002.1028437

[22] A. K. Maji, S. Mitra, B. Zhou, S. Bagchi, and A. Verma, "Mitigating interference in cloud services by middleware reconfiguration," in *Proceedings of the 15th International Middleware Conference*. ACM, 2014, pp. 277–288.

[23] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," in *ACM SIGOPS Operating Systems Review*, vol. 35, no. 5. ACM, 2001, pp. 103–116.

[24] X. Zhu, D. Young, B. J. Watson, Z. Wang, J. Rolia, S. Singhal, B. McKee, C. Hyser, D. Gmach, R. Gardner *et al.*, "1000 islands: Integrated capacity and workload management for the next generation data center," in *Autonomic Computing, 2008. ICAC'08. International Conference on*. IEEE, 2008, pp. 172–181.

[25] P. Xiong, C. Pu, X. Zhu, and R. Griffith, "vperfguard: an automated model-driven framework for application performance diagnosis in consolidated cloud environments," in *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*. ACM, 2013, pp. 271–282.

[26] N. Tran and D. A. Reed, "Automatic ARIMA time series modeling for adaptive I/O prefetching," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, no. 4, pp. 362–377, 2004. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TPDS.2004.1271185

[27] M. Li, D. Ganesan, and P. J. Shenoy, "PRESTO: feedback-driven data management in sensor networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 4, pp. 1256–1269, 2009. [Online]. Available: http://doi.acm.org/10.1145/1618562.1618581

[28] B. Zhu and S. Sastry, "Revisit dynamic ARIMA based anomaly detection," in *PASSAT/SocialCom 2011, Privacy, Security, Risk and Trust (PASSAT), 2011 IEEE Third International Conference on and 2011 IEEE Third International Conference on Social Computing (SocialCom), Boston, MA, USA, 9-11 Oct., 2011*, 2011, pp. 1263–1268. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/PASSAT/SocialCom.2011.84

[29] P. Cortez, M. Rio, M. Rocha, and P. Sousa, "Multi-scale internet traffic forecasting using neural networks and time series methods," *Expert Systems*, vol. 29, no. 2, pp. 143–155, 2012.

[30] J. Li and A. W. Moore, "Forecasting web page views: methods and observations," *Journal of Machine Learning Research*, vol. 9, no. 10, pp. 2217–2250, 2008.

[31] M. Couceiro, P. Romano, and L. Rodrigues, "A machine learning approach to performance prediction of total order broadcast protocols," in *4th IEEE SASO*, 2010, pp. 184–193.

[32] M. Stokely, A. Mehrabian, C. Albrecht, F. Labelle, and A. Merchant, "Projecting disk usage based on historical trends in a cloud environment," in *Proceedings of the 3rd workshop on ScienceCloud*, 2012, pp. 63–70.

[33] N. K. Ahmed, A. F. Atiya, N. E. Gayar, and H. El-Shishiny, "An empirical comparison of machine learning models for time series forecasting," *Econometric Reviews*, vol. 29, no. 5-6, pp. 594–621, 2010.

[34] L. M. Saini and M. K. Soni, "Artificial neural network-based peak load forecasting using conjugate gradient methods," *Power Systems, IEEE Transactions on*, vol. 17, no. 3, pp. 907–912, 2002.

[35] S. F. Crone, M. Hibon, and K. Nikolopoulos, "Advances in forecasting with neural networks? empirical evidence from the nn3 competition on time series prediction," *International Journal of Forecasting*, vol. 27, no. 3, pp. 635–660, 2011.

[36] K.-L. Ho, Y.-Y. Hsu, and C.-C. Yang, "Short term load forecasting using a multilayer neural network with an adaptive learning algorithm," *Power Systems, IEEE Transactions on*, vol. 7, no. 1, pp. 141–149, 1992.