CPU Workload forecasting of Machines in Data Centers using LSTM Recurrent Neural Networks and ARIMA Models

Deepak Janardhanan
College of Engineering and Informatics
National University of Ireland, Galway
Galway City, Ireland
Email: djdeepakjana72@gmail.com

Enda Barrett
College of Engineering and Informatics
National University of Ireland, Galway
Galway City, Ireland
Email: enda.barrett@nuigalway.ie

Abstract—The advent of Data Science has led to data being evermore useful for an increasing number of organizations who want to extract knowledge from it for financial and research purposes. This has triggered data to be mined at an even faster pace causing the rise of Data Centers that host over thousands of machines together with thousands of jobs running in each of those machines. The growing complexities associated with managing such a huge infrastructure has caused the scheduling management systems to be inefficient at resource allocation across these machines. Hence, resource usage forecasting of machines in data centers is a growing area for research. This study focuses on the Time Series forecasting of CPU usage of machines in data centers using Long Short-Term Memory (LSTM) Network and evaluating it against the widely used and traditional autoregressive integrated moving average (ARIMA) models for forecasting. The final LSTM model had a forecasting error in the range of 17-23% compared to ARIMA model's 3742%. The results clearly show that LSTM models performed more consistently due to their ability to learn non-linear data much better than ARIMA models.

Keywords—LSTM Network, ARIMA Model, CPU Usage forecasting, Data Center

I. INTRODUCTION

One of the major reasons for service disruptions in data centers is the inefficient allocation of work to machines, leading to resources not being utilized fully. A typical modernday machine in a data center can run hundreds to thousands of jobs per day with fluctuating resource requirements. Every data center has a management system that schedules units of work to these machines, but such systems fail to prioritize and adapt to the growing scheduling complexities in data centers. The main reason being their inability to take into consideration a machine's current state (CPU and Memory usage, number of running jobs etc.) before scheduling.

It was noted from previous work [1] that machines in Google's data center use on average only 45% of its CPU resources. The same number is around 60% for memory usage. On another note, Google's data centers all around the world use up 0.01% of the global power which is enough to light up 200,000 homes [2]. Even a 1% increase in efficiency of Google data centers can power over a thousand homes. These statistics are of just Google's Data Centers. Energy efficiency can be maximized with the use of Scheduling Systems that model a machines behavior before scheduling work on them. This work aims to provide a research contribution towards building scheduling systems that manage resources more efficiently and dynamically in data centers.

This study focuses on forecasting CPU usage of machines in Data Centers. In our approach, separate time series detailing CPU usage of machines are extracted from Google's cluster dataset. Later, exploratory time series analyses are carried out to understand the nature of the time series and two models of varying architecture are used to solve the forecasting problem. Firstly, the ARIMA model which is a proven traditional method for time series forecasting is used. Secondly, LSTM Recurrent Neural Network whose architectural design is very much suited for time series forecasting is used. The models are then evaluated and compared detailing the differences in the two models. The study used the cluster data trace released by Google that details the monitoring and resource usage information of a cluster of machines for a 29-day trace period. There isn't any other publicly available dataset that details scheduling information of a cluster as complex and large as this. The dataset is over 300GB in size and consists of scheduling information of over 12000 machines scattered across 5 tables, some of which span over a billion rows.

The rest of the paper is organized as follows. Section 2 describes the related work in the field. In Section 3, the background required to understand time series forecasting is explained including ARIMA models and LSTM Recurrent Neural Network. Section 4 and 5 details the feature extraction steps carried out and the exploratory analysis respectively. The modelling process using ARIMA and LSTM Models are detailed in Sections 6 and 7 respectively. Section 8 explains the evaluation of the two models with final conclusions of the research explained in Section 9 and possible future work described in Section 10.

II. RELATED WORK

Di et al. [3] presents the design of a prediction algorithm based on a Bayes Model to estimate mean load in machines over a future time interval (up to 16 hours). The data is first reduced to a set of mean load predictions each starting from the current time and using Bayes model with 9 features detailing the current state of the machine. They mention that their model improved load prediction accuracy by 5% compared to other state of the art methods that are based on moving averages, auto-regression and noise filters. The time series method used in this approach is the aggregation of CPU usage into larger time windows to facilitate better forecasting. Since this leads to significant loss

978-1-908320/93/3/$31.00©2017 IEEE

of information, our study aims to forecast CPU usage for every 5 minute windows and incorporate more granular forecasts.

Ismeel and Miri [4] discuss developing a prediction model using k-means clustering and Extreme Learning Machines (ELMs) to estimate future Virtual Machine (VM) requests in data centers. The methodology first creates various categories of VM clusters with an end goal of developing a prediction model for each of these clusters. K-means clustering is used to create clusters of similar VM. Then for each cluster an ELM workload forecasting model is designed to estimate the number of requests for each cluster that may arrive in future time periods. The study evaluates its model on the Google's cluster-usage traces with their results suggesting that their model outperforms other similar models. While this approach aims to forecast VM requests, our study focuses on CPU usage estimation that are more dynamic and granular in nature.

Cao et al. [5] showcase their work using an Ensemble method to gain better performance in CPU load predictions in a cloud environment. The proposed model comprises of 2 layers, one for predictor optimization used for incorporating new predictor instances and removes instances with inferior performance. Another set of ensemble layers are used to facilitate providing feedback to the previous layer to adopt appropriate optimization strategies. This is achieved using a scoring method that assigns each predictor with a score representing its performance which is periodically updated. The study concludes that an ensemble model exhibits better forecasts and stability than individual models like Autoregression and Exponential smoothing. Our study uses the LSTM network to understand its forecasting prowess when compared to the models in this study.

Many of these previous works studied workload prediction of machines for large time windows. As mentioned earlier, this study aims to understand the growing complexities in data center resource allocation by building forecasting models around a granular time series dataset wherein CPU usage forecasts are generated for 5-minute time windows. More generally, previous work [6] [7][8][9][10] has also examined scheduling approaches using methods such as Reinforcement Learning and Genetic Algorithms, focussing not strictly on predicting CPU but also on optimising application response time, workflow scheduling and predicting network bandwidth. The approach detailed in this work can be applied in these more general settings and we hope will form the basis for further additional work.

III. TIME SERIES FORECASTING

A time series refers to a sequence of observations that make a record of a particular activity over a period of time [11]. Due to the observations being sampled across time, there is an introduction of correlation between these points. Forecasting an activity or an observation usually involves collating its historical time series and finding patterns in them. Defining properties of time series data like trends and seasonal patterns can be identified from a time series plot and amount to the first step in any time series forecasting problem. Generally, transforming a time series into a smoothed-out version and then plotting it can reveal patterns that were previously unknown. One such commonly used smoothing technique is the Moving Average. The moving average method is used for measuring seasonal variations in a time series by computing the arithmetic mean of values for time intervals throughout the time series and reducing any volatility in the data.

To further understand randomness in the data, the concept of Stationarity is introduced. A time series is said to be strictly stationary if the probability distribution of a set of values in that time series remains the same even when the set is shifted in time. Many of the real-world time series data are non-stationary in nature and hence require data adjustments to make them stationary by removing trends that are increasing or decreasing in nature. One such commonly used approach is differencing the time series. The difference operation is carried out by subtracting the current values with its past values. This is effectively called first order differencing. Similarly, differencing can be applied successively if required to further remove trends completely albeit at the cost of losing more and more information. Autocorrelation Function (ACF) of a time series is the amount of correlation that current values have with past values. ACF and Partial-ACF (PACF) plots help identify any seasonal patterns in the data.

A. ARIMA MODELS

White noise is a time series where each observation is randomly drawn from a population with variance and mean equal to zero. Typically, time series are required to follow such a pattern for better forecasting and any deviation from this amounts to a violation. Autoregressive(AR) and Moving Average(MA) are 2 models that help rectify these violations if identified. A stationary series is said to have no trend with constant variations in mean and fluctuates with a consistent pattern. This means that the autocorrelation would remain constant and form a consolidation of signal and noise. ARIMA models can be used to act as a filter that separates the signal and noise after which the signal is used to forecast values in the future. Forecasting using ARIMA is via an equation that is linear, where the predictors are lags of dependent variables and/or the errors (forecast errors). The model with only lag values of the variable becomes an autoregressive model. Hence, coefficients used in ARIMA consist of lagged errors that are optimized using nonlinear optimization [12]. In short an ARIMA model is denoted by ARIMA(p,d,q) where: p is the total autoregressive terms i.e AR terms, d is the total differences needed for stationarity and q is the total lagged forecast errors while forecasting i.e MA terms. Identifying the relevant ARIMA model for a given time series involves determining the values of p, d and q. To determine if AR and MA terms are necessary, ACF and PACF plots are used.

B. LSTM NETWORKS

Artificial neural networks are a computational paradigm that mimic biological neural networks through a network of connected computational units called neurons that are organized as layers. A Recurrent Neural Network (RNN) follows such a paradigm that is designed specifically for analyzing information where there is dependence between the current and previous values. LSTM networks are a class of RNN that are used to interpret and learn even long-term dependencies [13]. The major feature associated with LSTM networks are their ability to retain and persist information for long sequences or periods of time.

LSTM networks have 4 processing units within a single repeating unit as shown in Fig. 1. The line that runs through the repeating modules in the diagram is the cell state that runs through the entire LSTM network modifying information as it traverses. The pink blocks in the figure represent modules called gates that modify the cell state optionally.

For every input X the LSTM decides the amount of information that needs to be removed from the previous cell state using a sigmoid layer, also called the Forget layer. Here, C represents the cell state of the current repeating module which is passed as input to the next repeating module. For each cell state, C(t-1) received by the current cell, the forget layer outputs a value between 0 and 1 representing the amount of information to be removed from the cell state.

The LSTM network then decides the amount of new information that needs to be included in the cell state. This has two parts: a *sigmoid* layer and a *tanh* layer. The output of these two layers are multiplied and added to the cell state computed by the forget layer. Finally, each cell needs to output a value. This is done using another sigmoid and tanh layer which outputs a filtered down version of the cell state. LSTM networks as a result are well suited for time series forecasting as it retains information that govern the current state of an activity.

IV. FEATURE EXTRACTION: GOOGLE'S CLUSTER DATASET

The cluster data released by Google consists of 29-day worth of workload information of 12,453 machines scattered across 5 different tables. The *Machine Events* table describes machine states and hardware specifications over the trace period. The *Machine Attributes* table describe machine properties such as kernel version, clock speed etc. over the trace period. Life cycle of a job from start to finish i.e whether it's waiting for resource, failed, finished, lost and evicted is detailed in the *Job Events* table. Similarly *Task Events* table (over 100 million records) details the life cycle of a task from start to finish and finally the *Task Resource Usage* table (over 1 billion records) contains the resource (CPU, Memory, Disk I/O etc.) consumption of tasks over the trace period at every 5-minute intervals.

BigQuery is a Big Data querying tool in the Google Cloud Platform that is used to facilitate SQL querying on large datasets. Once imported, large aggregation such as cross joins across tables can be executed in very less time. Using the BigQuery platform, a time series dataset is created detailing CPU Usage of machines for every 5-minute interval, creating a total of 8352 data points for the 29-day period. The dataset is processed from the Machine Events, Task Events and Task Usage tables of Google's cluster dataset. To extract the CPU workload of machines in each time window accurately, we take into account the fact that some tasks run only partially in some 5-minute windows. Hence for each window, CPU is computed by summing all the task's CPU readings and then multiplying each reading with a weight equal to the amount of overlap that it has in the window. This ensures that a task with a one second run time in a 5-minute window is not counted as the task running for entire window. There were instances where a period of inactivity in some machines were observed wherein N/A values were present in the CPU usage feature. Replacing them with 0 value is not characteristic of a machine. Hence, a linear interpolation was used that filled the N/A values with replacements which linearly interpolated neighboring values to maintain continuity in the series.

V. EXPLORATORY ANALYSIS

Its impractical to manually build a model for all 12000+ machines in the dataset. Explained here in detail are the time series analysis, modelling and forecasting approaches conducted for one machine (ID: 979583) and the same was followed for other machines too, though just briefly generalized and mentioned. The most active machines in the dataset were chosen for this study as they have a CPU usage value throughout the trace period and minimal N/A values. Henceforth, the machine with ID 979583 is referred to as Machine A. Time series plots of Machine A (see Fig. 2) exhibit daily seasonal patterns with sinusoidal fluctuations every day. A major difference between different machines modelled in our study is the maximum level of CPU usage reached. Machine A has a maximum usage value of 0.4546, while the same for other machines were significantly different.

VI. ARIMA MODELLING

In order to assess stability in the data, moving average summary points were computed for multiple values of time period which potentially smoothed out the raw data. After various considerations, a 2.5 hour moving average smoothing was chosen which brought in stability to the time series without losing too much information. This transformation on the time series was same for ARIMA models for all machines. The Augmented Dickey Fuller (ADF) test were used to determine whether a change in future values in a time series is dependent on previous lagged data and a linear trend. The Null hypothesis of ADF test represents a non-stationary time series. Other tests like ACF and PACF plots were conducted to visually inspect stationarity in a series. From the ACF plots it's concluded that there are significant autocorrelations with numerous lags for machine A. As a result, first order differencing was carried out. After differencing, same tests were again carried out and found Machine A time series to be stationary.

After careful considerations and combination of all parameters, a final ARIMA (2,1,2) model was fit on the Machine A Time series with seasonal components included in our model. The final ARIMA model for Machine-A had an RMSE of 0.0078 on the training data and 0.091 on the test data.

The forecasts (see Fig. 3) were able to follow patterns similar to the original time series till time 200 with clear mirroring of the slightest variations in the actual data. But after time 200, although it keeps up with the sinusoidal variations in the data, it can't mimic it completely. Hence, ARIMA modelling is a good forecasting model for near time predictions but fails to predict

978-1-908320/93/3/$31.00©2017 IEEE

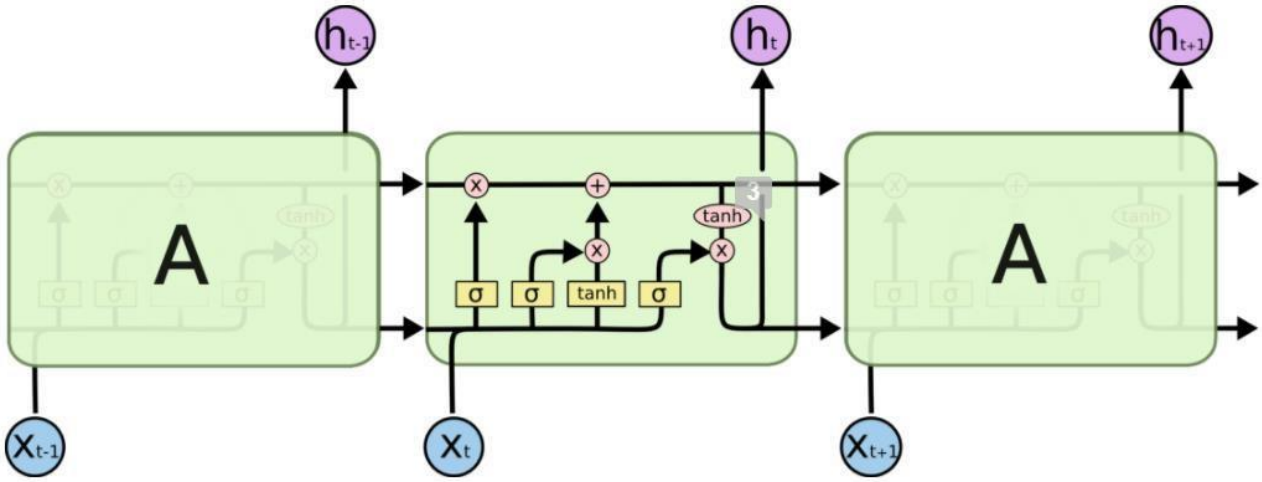

Figure 1. Internal structure of a Long Short-Term Memory Network

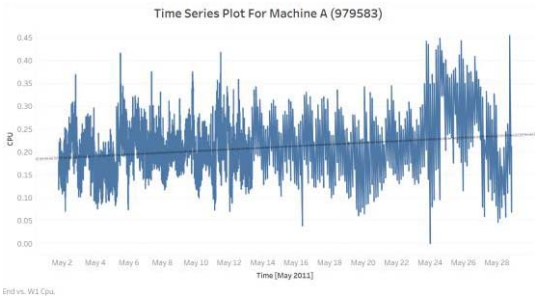

Figure 2. Time Series plot of Machine A

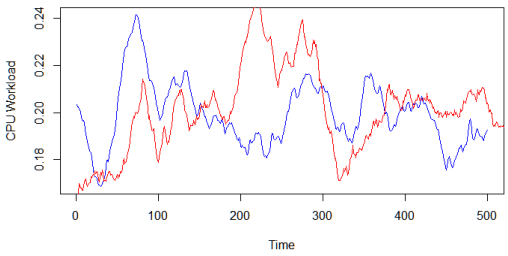

Figure 3. Forecasts (Red) of ARIMA (2,1,2) model with seasonal components included VS Actual (Blue) Time Series of Machine-A

long term CPU usage values. The results of ARIMA models for other machines are also similar where the forecast follows the same pattern as the actual data for near time predictions. But the models lose credibility in making long term predictions. Note that the y-axis scale of the plot in Fig. 3 is in the range 0.2 and 0.5, meaning the forecasts are accurate but due to granularity of the data, it's cannot be interpreted visually with full scale plots.

VII. LSTM NETWORK MODELLING

Firstly, the time series were transformed to a stationary time series. From the previous section, it was found that performing a first order differencing was sufficient to stabilize the data without losing much information. Hence, first order differencing was applied. The activation function of LSTM is a tanh function which can output a value between -1 to 1. Hence the time series is transformed by normalizing it within this range before splitting it into training and testing sets. Hyperparameters were tuned after training multiple LSTM Networks with different combinations of training epoch and number of neurons in the hidden layer. Firstly, the effects on training losses were assessed for different training epochs numbers. Then, LSTM models were fit with varying number of neurons and their RMSE and MAPE values were calculated. After tuning the LSTM Network's hyperparameters, a final LSTM model configuration that yielded the best results when configured to 1 hidden layer with 5 neurons and training epochs set to 3000. The models were then fit to the training data and the forecasting approach was tweaked to facilitate better forecasts. Instead of forecasting each time step in the future from the test data, a more dynamic approach was used wherein we updated the model after forecasting each value in the series to include the new forecasted value too. The final LSTM Model for Machine A had a training RMSE value of 0.0317 and test RMSE of 0.0381. Fig. 4 shows the initial few forecasts made by the model.

To look closely at the forecasts, plots of initial sets of forecasts for Machine A were plotted. Specifically, CPU forecasts of the first 2, 12 and 24 Hours in the test data were plotted (see Fig. 4) against their actual CPU values to see how closely the model understood recent data (i.e training data). From plots (ii) and (iii) in Fig. 4, it can be noted that forecasts closely followed the actual CPU usage reading, displaying similar seasonal patterns to an extent. Plot(i) shows the first 2-hour forecasts, and it's evident that the model has trained on the training dataset well. As mentioned earlier, fine tuning the hyperparameters not only increased the performance with respect to metrics RMSE and MAPE, it also has been able to pick up seasonal components in the time series.

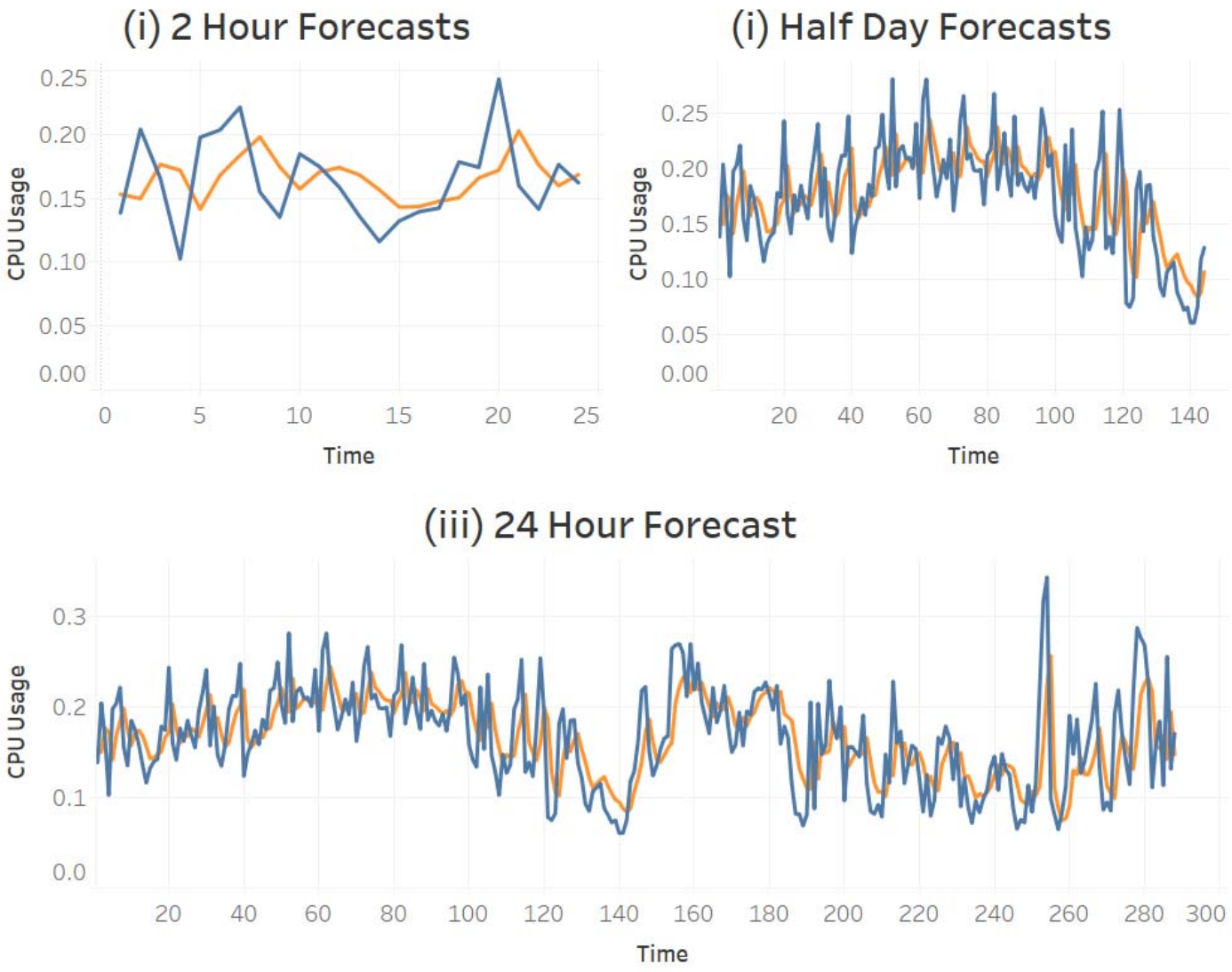

Figure 4. First (i)2 Hour, (ii)Half Day (iii)24 Hour Forecasts of LSTM Model for Machine A test data. Blue: Actual CPU vs Red: Forecasted CPU

The results of the LSTM models for machines other than Machine A exhibited similar model features. Time series of many machines were similar in nature hence didn't require much individual attention. For machines whose CPU usage time series were significantly different were modelled accordingly but exhibited just slightly fluctuating results. But one aspect that was found similar for all the models was the hyperparameters that were required to train them. As noticed in Machines A, there wasn't any notable change in the training RMSE scores when trained on 2000 and 3000 epochs. The number of neurons too were limited to 5 which gave the best results during training of models with a single hidden layer.

VIII. MODEL EVALUATION

Both the models were trained differently. Differencing and statistical moving average data transformations were used while modelling using ARIMA models. The final ARIMA (2,1,2) models for Machine A had good RMSE scores but significantly different test RMSE. The Akaike information criterion and Bayesian information criterion scores obtained in the ADF tests were also used to find the best ARIMA models. Both these metrics are found to have very high possibility of overfitting for AR processes [14]. CPU usage of a machine in data centers always have dependence between the previous and current values. This fundamentally makes a CPU usage time series an AR process. LSTM Network essentially being a non-linear time series model performed better than ARIMA model for almost all combinations of its hyperparameter.

ARIMA model was observed to be prone to overfitting for multiple CPU time series of machines. They performed well in terms of training RMSE values but poorly on the test data. On the other hand, LSTM models for all machines were consistently performing better without any significant fluctuations in model performance.

IX. CONCLUSION

The forecasting of CPU workload of Machines in Data Centers using LSTM Networks has shown significant improvement over the ARIMA models. To the author's knowledge, this is the first time LSTM networks has been used for forecasting CPU workload of machines in Data Centers. More specifically, this is the first piece of work that evaluated LSTM Network in its efficiency to forecast multiple highly volatile CPU usage time series extracted from Google's Cluster Data.

The problems associated with instability and fluctuations in CPU usage of machines in Google's data center and how they affect a time series forecasting model's accuracy are highlighted. The ARIMA modelling approach to forecasting CPU workload is detailed along with its limitations. More generally, its inability to model an unstable and volatile time series is explained taking the CPU usage time series as an example to showcase the level of transformations required to build a decent forecasting model out of it. The initial ARIMA models built couldn't fit a time series with high CPU fluctuations observed on an hourly basis. Significant data transformations have added little to no major improvements in the forecasting performance of the ARIMA model. The ARIMA models were evaluated and found to be overfitting the time series test data with significant difference in results between the RMSE scores for the training and testing phases of the model. The final ARIMA(2,1,2) model had a forecasting error in the range of 37.331 to 42.881% across multiple machines.

On the other hand, the LSTM Network forecasted the CPU workload with a forecasting error rate in the range of 17.566 to 23.65% for multiple machines. The use of LSTM Networks in this study is a new step towards estimation of complex workload of machines in data centers due to its ability to maintain memory of patterns and trends. In the end, it forecasted CPU workload better than many traditional forecasting models as well as some simpler versions of Recurrent Neural network. Hence, a case for LSTM Networks as an excellent time series forecasting tool for workload estimation of resources in data centers has been put forth in this study.

X. FUTURE WORK

This study detailed the time series forecasting process employed for univariate time series forecasting of CPU workload of machines in data center. As future work, improving the forecasting performance of the LSTM model can be carried out. One such approach can be the use of multiple predictor variables in learning and fitting a LSTM model around it. Some of the predictor variables that can be used are: number of tasks running in machines, the memory usage etc. Other features like the number of tasks failed, completed, restarted and killed in every 5-minute window could be extracted from the dataset too. There are other unconventional features that aren't used like resource usage metrics such as CPI (cycles per instructions), mean disk I/O time and memory accesses per instruction.

Another approach to forecasting the same data is by considering the time series as a sequence. Till now almost all Time Series forecasting approaches that have modelled this data have fit their models considering observations at each time step as a single input. CPU usage in this data had clear daily seasonal patterns. Hence considering an entire day's observations as a single data point i.e 288 features (1 day has 288 5-minute windows), where each time window of the day is considered as a feature. Such a dataset would basically be a sequence to sequence Time series forecasting problem where the model forecasts the CPU usage for the next day. Such an approach to time series forecasting especially on this data can be promising.

REFERENCES

[1] Charles Reiss et al. "Towards understanding heterogeneous clouds at scale: Google trace analysis". In: *Intel Science and Technology Center for Cloud Computing, Tech. Rep* 84 (2012).

[2] *Big Data and Data Center Fun Facts - AIS*. http://www.americanis.net/2014/big-data-data-center-fun-facts/. (Accessed on 09/30/2017).

[3] Sheng Di, Derrick Kondo, and Walfredo Cirne. "Host load prediction in a Google compute cloud with a Bayesian model". In: *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society Press. 2012, p. 21.

[4] Salam Ismaeel and Ali Miri. "Using ELM techniques to predict data centre VM requests". In: *Cyber Security and Cloud Computing (CSCloud), 2015 IEEE 2nd International Conference on*. IEEE. 2015, pp. 80–86.

[5] Jian Cao et al. "CPU load prediction for cloud environment based on a dynamic ensemble model". In: *Software: Practice and Experience* 44.7 (2014), pp. 793– 804.

[6] Martin Duggan et al. "A network aware approach for the scheduling of virtual machine migration during peak loads". In: *Cluster Computing* (2017), pp. 1–12.

[7] Martin Duggan et al. "An Autonomous Network Aware VM Migration Strategy in Cloud Data Centres". In: *Cloud and Autonomic Computing (ICCAC), 2016 International Conference on*. IEEE. 2016, pp. 24–32.

[8] Enda Barrett, Enda Howley, and Jim Duggan. "Applying reinforcement learning towards automating resource allocation and application scalability in the cloud". In: *Concurrency and Computation: Practice and Experience* 25.12 (2013), pp. 1656–1674.

[9] Martin Duggan et al. "A reinforcement learning approach for the scheduling of live migration from under utilised hosts". In: *Memetic Computing* (2016), pp. 1– 11.

[10] Enda Barrett, Enda Howley, and Jim Duggan. "A learning architecture for scheduling workflow applications in the cloud". In: *Web Services (ECOWS), 2011 Ninth IEEE European Conference on*. IEEE. 2011, pp. 83–90.

[11] Rainer Schlittgen. "Robert H. Shumway and David S. Stoffer: Time series analysis and its applications with R examples, 2nd edn." In: *AStA Advances in Statistical Analysis* 92.2 (2008), pp. 233–234.

[12] Robert Nau. "Statistical forecasting: notes on regression and time series analysis". In: *Durham: Fuqua School of Business, Duke University* (2015).

[13] *Understanding LSTM Networks – colah's blog*. http://colah.github.io/posts/2015-08-Understanding-LSTMs/. (Accessed on 09/30/2017).

[14] Yamei Liu. "Overfitting and forecasting: linear versus non-linear time series models". In: (2000).