Proceedings of the Federated Conference on Computer Science and Information Systems pp. 115–118
DOI: 10.15439/2020F213
ISSN 2300-5963 ACSIS, Vol. 21

Deep Bi-Directional LSTM Networks for Device Workload Forecasting

Dymitr Ruta
EBTIC, Khalifa University, UAE
dymitr.ruta@ku.ac.ae

Ling Cen
EBTIC, Khalifa University, UAE
cen.ling@ku.ac.ae

Quang Hieu Vu
Zalora, Singapore
quanghieu.vu@zalora.com

***Abstract*—Deep convolutional neural networks revolutionized the area of automated objects detection from images. Can the same be achieved in the domain of time series forecasting? Can one build a universal deep network that once trained on the past would be able to deliver accurate predictions reaching deep into the future for any even most diverse time series? This work is a first step in an attempt to address such a challenge in the context of a FEDCSIS'2020 Competition dedicated to network device workload prediction based on their historical time series data. We have developed and pre-trained a universal 3-layer bi-directional Long-Short-Term-Memory (LSTM) regression network that reported the most accurate hourly predictions of the weekly workload time series from the thousands of different network devices with diverse shape and seasonality profiles. We will also show how intuitive human-led post-processing of the raw LSTM predictions could easily destroy the generalization abilities of such prediction model.**

***Index Terms*—Workload prediction, time series, Long Short-Term Memory (LSTM), ensemble averaging**

I. Introduction

PREDICTIVE analytics on workload-related characteristics has become increasingly important. Reliable workload prediction of monitored devices becomes critical in order to proactively manage the capacity of connected infrastructure, mitigate cyber security risks and simply respond early to the anomalous behaviour of the monitored IT infrastructure [1]. Accurate forecasting of the future host workload plays also a central role for robust scheduling and resources management in data centers and cloud computing and among many expected benefits could lead to reduced operational cost, for example in a form of eliminated or cut idle time of the devices [2], [3], [4].

Prediction of future workload characteristics has received considerable research interests in both academia and industrial applicationss. Simple linear techniques like (auto-regressive) moving average (ARMA) models have been used heavily in this field [5], [6], [7], and enjoyed relatively good performance at the very low computational cost. As the complexity of the time-series increases, a subtle dependence of the future on the past may be non-linearly implied in the uni- or multi-variate series and linear models struggle or completely fail to efficiently unscramble such dependence. In recent years, workload prediction has also been attempted using non-linear machine learning models, e.g. Bayesian model [8], neuro-fuzzy and Bayesian inference [9], Neural Network (NN) [10]. Despite the fact that these non-linear models are not particularly suited to learn temporal dependencies in sequences, their strong regression capabilities often led to the predictive performance improvement measured in the static conditions shifted over moving window. The developments of the Recurrent Neural Networks (RNN) managed to better capture internal correlations along the data series and in an instant became naturally suited and applied to to sequential data analysis, including workload prediction. In [2], an adaptive model was developed for highly-variable workloads prediction by integrating a Top-Sparse Auto-Encoder (TSA) and Gated Recurrent Unit (GRU) blocks into RNN. In [3], workload sequences in Cloud and Grid systems were predicted by developing a model of stacking prediction algorithms using RNN and Autoencoder. An approach based on the Long Short-Term Memory (LSTM) encoder-decoder network with attention mechanism was proposed in [11]. In [12], a GRU-based encoder-decoder network containing 2 gated recurrent neural networks was implemented for prediction of multi-step-ahead host workload in cloud computing.

Accurate workload prediction is a challenging problem. Different time series from various devices typically have varied patterns that not only lack of well pronounced stationarity but also are often full of sudden spikes, dropouts, staircase and other complex temporal patterns. This makes them very difficult to model and predict using the same predictor or even the same class of predictive models. This paper presents an ensemble model based on Bi-Directional Long Short-Term Memory (BiLSTM) networks developed and pre-trained for prediction of a large class of network device workload time series. At its core we have proposed a regression network with our own architecture containing 3 BiLSTM layers that appears to perform very well for a very diverse workload time series. Its prediction accuracy evaluated on thousands of different devices' workload series was acknowledged to generate best results in the FedCSIS'2020 Data Mining Challenge, details of which are further elaborated in sections below.

The remainder of the paper is organized as follows. The FedCSIS'2020 Challenge is briefly described in Section II. Data pre-processing is presented in Section III, followed with the description of the core BiLSTM network and its ensemble aggregation model in Section IV and the experimental results in Section V. Concluding remarks are provided in Section VI.

II. FEDCSIS'2020 CHALLENGE

The aim of this challenge was to predict workload-related characteristics of monitored devices based on historical data collected from these devices. Accurate prediction of device workload can be quite useful to manage infrastructure capacity, mitigate cyber security risks and early respond to anomalous events. The data provided in the competition were collected by EMCA Software that is a Polish vendor of Energy Logserver, a globally operating system capable of collecting data from various log sources to provide in-depth data analysis and alerting to its end-users [1].

The training data were organized in hourly aggregated values of various workload characteristics extracted from device logs, provided in the form of a CSV table also containing device identifiers and timestamps of the aggregation windows. Overall there were 24249 devices' time series, each containing 7 sets of workload related statistics including mean, standard deviation and the candlestick intra-hour aggregates of open, high, low, close of each hour. Each of such multivariate time series was captured along 1924 subsequent hours spanning over 80 days from 2019-12-2 07:00:00 to 2020-2-20 10:00:00, however, a significant number of values were missing.

Based on these data the task of the competition was to predict the following week i.e. a 168-hourly future sequences starting at 2020-02-20 11:00:00 - directly after the end of the training data, of the mean workload for only the selected subset of 10000 series. The competitors' solutions provided in a form of 10000 168-element vectors y_i were evaluated against the true values f_i using the R^2 score defined as follows:

$$R^2(y, f) = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - \overline{y})^2} \tag{1}$$

During the competition only partial feedback on the performance of the competitors' models was provided by the knowledgepit.ml platform, on which the competition was hosted. This feedback was in a form of the preliminary R^2 score that was computed over a small unknown subset of the complete testing set of 10000 series.

It is important to note that although $\overline{y}$ is supposed to stand for the mean of the true values of the testing series, in fact in this competition the scope of the mean has been extended to include a complete series, i.e. it is a mean of both training and testing parts of the series. This has been necessitated by the risk of infinite R^2 scores reported over the testing week that could easily happen for a single flat or dropped out signal that would hijack the complete score of 10000 series.

R^2 score scaled between $-\infty$ and 1 is considered to be a very "tough" or penalizing measure of the regression performance compared to the mean squared error (MSE) or relative MSE measurements. This is simply because it is open to the large negative scores observed for random predictions and to even reach the score of 0 one has to correctly match the mean between predicted and actual values.

III. DATA PROCESSING

Considering that the task of the competition concerned the prediction of only the mean values of devices' workload characteristics, there was a founded temptation to only use the mean values time series rather than the whole candlestick series and the signal volatility as features. Following a rapid prototyping experiments with a couple of standard and simple regression models we have concluded that none of the time series other than the mean, even in the multivariate regression setup, bring any visible improvement in predicting the future hourly mean values of the series, while their inclusion only multiplies the computational cost of the regression model. Backed up by these results we have reduced the data to include only the mean workload series for both training and testing sets. Our task was therefore simplified to predicting 168 subsequent values of the 10000 univariate time series based on their own history of 1924 hourly values as well as other univariate time series available in the training set containing in total 24249 series of 1924 hourly mean workload values. For computational simplicity the whole data were represented as a matrix $X^{[24249\times1924]}$ of values in a single precision format to reduce memory requirements.

A. Filling missing values

Out of the 46655076 values, 2265379, which is almost 5%, were missing. Since the models we intended to apply to the regression problem did not accept missing values we have developed a simple scheme to fill all missing values that simply fills the average values from the same hour in the same week-day across all weeks if there is at least one values given for this hour, otherwise, it is filled with the average of its closed hour, i.e. previous or next hour in the same weekday.

B. Normalization

After all missing values are filled, each time series is then normalized to zero mean and unit standard deviation, expressed as:

$$\tilde{x} = \frac{x - \mu}{\sigma}, \tag{2}$$

where μ and σ are the mean and standard deviation, yielding the output time series with

$$\tilde{\mu} = 0 \ , \ \tilde{\sigma} = 1. \tag{3}$$

C. Data partition

Competition participants are required to predict workload for only selected 10000 among the 24249 device time series. Accordingly, the whole data set X was partitioned into training and validation sets.

- The training set contains the selected 10000 time series, which will be used for model training and testing.
- The validation set contains all the remaining 14249 workload time series after ensuring that:
  - all of the values used for validation are positive;
  - all time series have positive standard deviation.

IV. The universal time series regression network

Note that after the preprocessing we are left with a set of time-aligned but independent series of data, each of which needs forward prediction in time. Using traditional time series approach one would in fact build a separate model for all of them independently and also independently use them to forecast the time series' future. The novelty of our proposed solution is that it intends to build a single model that is trained on all diverse time series and once pre-trained it is effectively expected to be able to predict the future of any other time-aligned time-series, based on its past, without any further (re)-training necessary. We have decided to build such universal time series prediction model using Long-Short-Term-Memory (LSTM) networks that are particularly suited for predicting deep futures of the variety of diverse time series data.

A. Long-Short-Term-Memory networks

LSTM networks are powerful family of models based on deep recurrent learning regression networks that are very flexible with a freedom of layered architecture design and powerful gated mechanism of LSTM layers that give them the ability to manipulate its memory state to extract complex patterns over long sequentially arranged input feature space. Due to these features LSTMs are known to successfully capture multitude of seasonalities, autocorrelations and other subtle time dependencies in both uni or multivariate mode and are reported to maintain stable accurate forecast deep into the future. As per the application recommendations we used bi-directional version of the LSTM (BiLSTM), that can learn from both past and future, which is suited to our problem when the predicted series is long and therefore has enough space to accommodate forward and backward learning patterns.

B. BiLSTM network architecture

There has been an iterative refinement process of constructing the final BiLSTM network architecture [13]. This process was guided by the observation of improved predictions with an increasing number of hidden units and a number of BiLSTM layers. Expanding the network along this tendency had two issues, however. Firstly the computational cost and hence the time of training grew very quickly, exponentially if expanding both the number of layers and their sizes. The second drawback is, that unless validation set was perfectly representative, the expanded network showed the tendency of becoming over-trained very quickly, although without clear and reliable rule as to when is the best moment to stop training. On top of this, allowing frequent validation evaluation during training is very costly and additionally slows down the experimentation phase of the network build.

To address the above issues we have noticed that we can compensate the additional cost of expanding the network by reducing the training set down the the k-last weeks. This process brought significant performance benefits up to when we tried to shrink the training series below 4 last weeks indicating that on average there is no predictive gain from learning from more than 3 weeks back. Eventually, we have expanded the BiLSTM layers up to 504 (3×168) hidden units and included 3 BiLSTM layers followed with 10% dropout layers. We have also tried to include ReLu layers that eliminate negative signals but eventually their impact turned out not to influence the results hence we dropped them. The 3 rounds of BiLSTM and dropout layers followed with two sets of dense layers separated with another dropout layer before eventually reaching the final (MSE) regression layer. The final network architecture, with which we have generated the final predictions, is presented in Figure 1:

1	'Input'	Sequence Input	Sequence input with 1 dimensions
2	'BiLSTM_1'	BiLSTM	BiLSTM with 504 hidden units
3	'Dropout_1'	Dropout	10% dropout
4	'BiLSTM_2'	BiLSTM	BiLSTM with 504 hidden units
5	'Dropout_2'	Dropout	10% dropout
6	'BiLSTM_3'	BiLSTM	BiLSTM with 504 hidden units
7	'Dropout_3'	Dropout	10% dropout
8	'Dense_1'	Fully Connected	100 fully connected layer
9	'Dropout_4'	Dropout	10% dropout
10	'Dense_2'	Fully Connected	1 fully connected layer
11	'Regression'	Regression Output	mean-squared-error

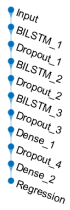

Figure 1. Deep BiLSTM Network Architecture

To take full advantage of the BiLSTM layers that require to look forward and backward in relation to the tested point we have trained the network in the sequence-to-sequence mode rather than the standard one-next-and-update mode. It is also worth noting that our validation strategy evolved from initially evaluating on the additional series not used in testing, through validating on the last 2 weeks of the available 10000 tested series, up to validating on just the last week of the available data. The training proceeded on the mini-batches of 64 randomly selected series and terminated when validation error has not been reduced within the last 50 iterations.

V. Experiments and post-processing

Once the network architecture has been established the experiments followed an iterative process of final parametric optimization guided by the regression performance measured on the validation set as indicated above. The generated validation and testing set predictions have been provisionally inspected particularly in terms of the signal and its trend continuity. For the vast majority of series that visually follow an established pattern the validation set predictions are very accurate as shown in Figure 2.

It is reassuring to observe a correct flat mean prediction whenever a signal resembles a random noise. Even unexpected sudden changes of the signal are to a certain extent reflected in the predictions. Overall the presented BiLSTM model in its standalone form received the preliminary R^2 score of 0.31

A. post-processing

Analysis of the predictions revealed some perceived issues for the time series that have sudden change of the signal near the end of the series as well as occasional signal dropouts to 0 or near 0. In response to these observations we have developed a set of additional post-processing techniques that were supposed to fix these issues.

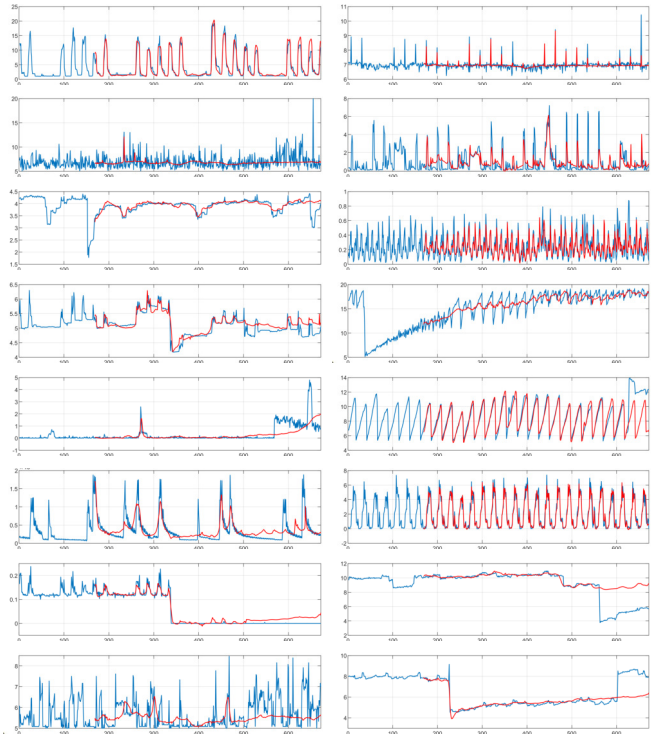

Figure 2. Validation set series samples (blue) and their predictions (red)

We have identified two categories of significant disparity between the last week of the training series and the following predictions that we have decided to address in post-processing:

- Signal dropouts: observed when all values of the training time series in the last week are (near) 0 but the predictions are not completely 0. In such case we have concluded that the signal which drops at its end to 0, should maintain its prediction also at the 0 level for the whole week.
- Excessive difference: observed when the mean difference between the last weeks' values and the predicted values is excessively large i.e. exceeds 3 standard deviations of the complete training series. In such case we have introduced two adjustments: moderation that simply computes the average between the last week and the predictions and trend alignment that replaces the network predictions with a simple average of the 3-weekly trend matched to continuously extrapolate the end of the series in the direction of the gradient between the last two weeks.

The above post-processing techniques have been applied to the predictions and their effects measured in a form of preliminary and, later revealed, final testing scores. Interestingly, all of these techniques improved the R^2 score but only reported over the validation set, also reflected by the gains in the preliminary scores. Unfortunately, as the final testing revealed after the end of the competition, none of the pre-processing techniques improved the performance measured over the complete testing set. The comparison of the preliminary and final testing R^2 scores are shown in the Table I.

Table I
PRELIMINARY AND FINAL R^2 SCORES OF THE BILSTM MODEL WITH VARIOUS POST-PROCESSING TECHNIQUES

model	post-proc	preliminary R^2	final R^2
BiLSTM	none	0.309	0.290
BiLSTM	dropout	0.312	0.288
BiLSTM	moderation	0.318	-2.38
BiLSTM	trend alignment	0.321	-0.779

B. Ensemble averaging

As a final step an arbitrary number (20-30) of the top solutions have been aggregated with the simple mean operator. Although this operation expectedly resulted with the average performance improvements, notably R^2 score was elevated to 0.322, these gains were negligible compared to the performance degradation caused by the signal post-processing.

VI. CONCLUSIONS

In this paper, we presented a powerful 3-layer BiLSTM network that has the capability to deliver deep predictions of a very wide and diverse spectrum of time series. The model achieved the performance of the R^2 score of nearly 0.3 beating all other competitive solutions in the FedCSiS'2020 competition. Although the subsequent post-processing introduced to the model turned out be a bad idea, this lesson learnt gives us confidence in the native capability of the deep BiLSTM networks to reliably predict diverse time series that the arbitrary and selective human corrections can only damage.

REFERENCES

[1] FedCSIS 2020 Challenge: Network Device Workload Prediction, *https://knowledgepit.ml/fedcsis20-challenge/*.
[2] Z. Chen, J. Hu, G. Min, A. Zomaya, and T. El-Ghazawi, "Towards Accurate Prediction for High-Dimensional and Highly-Variable Cloud Workloads with Deep Learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 4, pp. 923-934, April 2020.
[3] H. Nguyen, S. Woo, J. Im, T. Jun, and D. Kim, "A Workload Prediction Approach Using Models Stacking Based on Recurrent Neural Network and Autoencoder," *IEEE Int. Conference on High Performance Computing and Communications, IEEE International Conference on Smart City, IEEE International Conference on Data Science and Systems*, Dec. 2016.
[4] K. Qazi and I. Aizenberg, Towards quantum computing algorithms for datacenter workload predictions, *IEEE Int. Conf. on Cloud Comput.*, 2018.
[5] P. Saripalli, G. Kiran, R. Shankar, H. Narware, and N. Bindal, "Load prediction and hot spot detection models for autonomic cloud computing," *IEEE Int. Conf. in Utility and Cloud Computing*, pp. 397–402, 2011.
[6] R. Calheiros, E. Masoumi, R. Ranjan, R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS," *IEEE Trans. Cloud Comput.*, vol. 3, no. 4, pp. 449–458, 2014.
[7] P. Dinda, and D. O'Hallaron, "Host load prediction using linear models," Cluster Computing," vol. 3, no. 4, pp. 265–280, 2000.
[8] S. Di, D. Kondo, W. Cirne, "Host load prediction in a Google compute cloud with a Bayesian model," *Proc. of IEEE Int. Conf. on High Performance Computing, Networking, Storage and Analysis*, 2012.
[9] F. Benhammadi, Z. Gessoum, A. Mokhtari, CPU load prediction using neuro-fuzzy Bayesian inferences. Neurocomputing 74, 1606–1616 (2011)
[10] J. Kumar, A. Singh, Workload prediction in cloud using artificial neural net. and adaptive diff. evolution, *Futur. Gen. Comput. Syst.* 81:41–52, 2018.
[11] Y. Zhu, W. Zhang, Y. Chen and H. Gao, "A novel approach to workload prediction using attention-based LSTM encoder-decoder network in cloud environment," *EURASIP Journal on Wireless Communications and Networking*, Article number: 274, 2019.
[12] C. Peng, Y. Li, Y. Yu, Y. Zhou and S. Du, "Multi-step-ahead host load prediction with GRU based encoder-decoder in cloud computing," *IEEE Int. Conference on Knowledge and Smart Technology*, pp. 186–191.
[13] S. Hochreiter, J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.