Machine Learning Model for GSM BSC Control Plane Units

Aswathy K, Kaustuv Saha, Pardhasaradhi R, Athi Narayanan

ABSTRACT--- At maximum traffic intensity i.e. during the busy hour, the GSM BSC signalling units (BSU) measured CPU load will be at its peak. The BSUs CPU load is a function of the number of transceivers (TRXs) mapped to it and hence the volume of offered traffic being handled by the unit. The unit CPU load is also a function of the nature of the offered load, i.e. with the same volume of offered load, the CPU load with the nominal traffic profile would be different as compared to some other arbitrary traffic profile. To manage future traffic growth, a model to estimate the BSU unit CPU load is an essential need. In recent times, using Machine Learning (ML) to develop such a model is an approach that has gained wide acceptance. Since, the estimation of CPU load is difficult as it depends on large set of parameters, machine learning approach is more scalable. In this paper, we describe a back-propagation neural network model that was developed to estimate the BSU unit CPU load. We describe the model parameters and choices and implementation architecture, and estimate its accuracy of prediction, based on an evaluation data set. We also discuss alternative ML architectures and compare their relative prediction accuracies, to the primary ML model.

Index Terms - machine learning, base station controller, neural networks, support vector regression, regression model

I. INTRODUCTION

GSM is the most popular second-generation cellular system. Base Station Subsystem (BSS) is the front-end of the network architecture, which manages the wireless connection between the UE and the network [1]. The Base Station Controller (BSC) performs radio resource, connection and mobility man- agement within the BSS. In the BSC, BSC signalling unit (BSU) controls or manages the signalling unit with the UE over the air interface channels associated with transceivers (TRXs) [14]. TRXs are the device which can transmit and receive data to and from the UE. Multiple TRXs are connected to a single BSU unit. For a given traffic profile, the offered load per BS is given by the "capacity per TRX * number of TRXs connected to the BSU" [13] [15].

From the above summary, it is clear that the CPU load of a BSU unit is a function of the traffic profile (among other things). The traffic profile in each deployment is unique. Since dimensioning plays an important role in the network planning in telecommunications, it is important to do the traffic analysis and optimization. It can be used in network resource allocations, network architecture comparisons and performance evaluations [1] [3]. Hence, it is quite critical to develop a model to accurately estimate the CPU load of the BSU [4] [5].

Nowadays, machine learning has been widely used in networking for prediction, classification problems and for automation [11]. The machine learning concepts can be ap- plied in networking applications such as detection of network anomalies, to anticipate anomalies before they become a problem, identification of the root cause of anomalies, regression based data-driven network optimization and prediction of future events, classification of network events, and learning to take actions on the network using Reinforcement Learning [2]. The basic step in a machine learning processes is shown in the figure Fig.1. They are dataset pre-processing, model creation and assessment. For achieving a better result, dataset should be appropriate and should be in a format for providing to a machine learning model. So, in dataset pre-processing step, the datasets are rescaled, standardized and normalized according to the dataset and needs. Next step in machine learning is the model creation. We need to choose a proper machine learning algorithm according to the dataset and the output we needed. Using that we need to create a model and should train the model using the dataset. Next phase in the machine learning is the assessment. Here we will test and evaluate the model, how accurate it is predicting, how better the algorithm is, how to minimize the error, etc.

In this paper, we describe a back-propagation neural net- work model that was developed to estimate the BSU units CPU load [12]. This work was done as part of an internship project at Nokia Solutions and Networks and hence, the model is specific to the Nokia BSC. However, the model should be generically extendible to BSCs from other vendors as well.

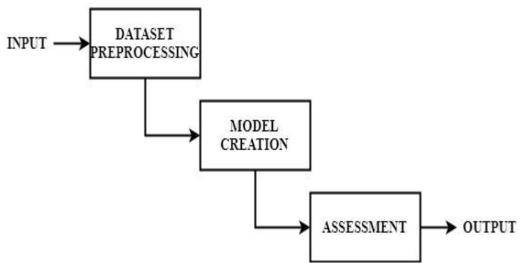

Fig. 1. Basic steps in machine learning.

Manuscript published on 30 August 2019.
***** Correspondence Author (s)

Aswathy K, Department of Mechanical Engineering Amrita Vishwa Vidyapeetham Amritapuri, India. (E-mail: achukumar001@gmail.com)
Kaustuv Saha, Nokia Solutions and Networks Pvt. Ltd. Bengaluru, India. (E-mail: kaustuv.saha@nokia.com)
Pardhasaradhi R, Nokia Solutions and Networks Pvt. Ltd. Bengaluru, India. (E-mail: pardhasaradhi.r@nokia.com)

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

The dataset we used here is the traffic profile parameters. Regression-based method in machine learning is used to create the model for estimating the CPU load.In next section we will be discussing about the dataset we used, the source of dataset and the features we took from the dataset. In section 3, we will be discussing about the main model we used for predicting the CPU load. In section 4, we discussed about the alternative models we used other than the main model. In section 5, we discussed about various results obtained from the main model and the alternative model. In the final section, we will be concluding our work and its future enhancements.

II. DATASET

The dataset comprises of a large number of traffic profile parameters. So, we selected a subset that influences the CPU load the most, as features for our model and following are those parameters:

- No. of Traffic Channel (TCH) allocations during a Mobile Originated Calls (MOC).
- No. of TCH allocations during a Mobile Terminated Calls (MTC).
- No. of successful allocations of Stand-alone Dedicated Control Channels (SDCCH) for location update.
- No. of SMSs successfully established on an SDCCH.
- No. of Mobile Originated (MO) SMSs successfully established on an SDCCH.
- No; of the successful internal intra-cell Hand overs (HO) on SDCCH or TCH.
- No. of the successful internal inter-cell HOs on SDCCH or TCH.
- No. of the conversation phases started.
- Average CPU load (in percentage).

The datasets are collected from two sources, first called dataset A which is from the labs and latter called dataset B from the customer. These datasets are generated in different time duration, hence the sampling frequencies are different, and so we normalized the frequencies. Number of samples in dataset A is nearly 180 and dataset B is nearly 20,000. We split both the datasets into three groups for training, validation and testing. We took 15 to 30 number of samples for testing and from the remaining, we took 80% for training and 20% for validation from the dataset A and B.

III. MAIN MODEL

The CPU load prediction model is made based on the back propagation neural network model. For each dataset, i.e., dataset A and dataset B, we made two different neural network architecture. In case of dataset A and dataset B, the neural network model we created contains four layers. First and last are the input and output layers in both the models and for each contains eight and one nodes respectively. In between those layers, two hidden layers are made, and contain ten number of nodes each in dataset A. In dataset B, two hidden layers are made, and contain fifteen number of nodes each.

In every layer in both the models, batch normalization is done [8]. We used tanh activation function in case of first model and in case of second model, linear activation functions are used [8].

The first model is being running for 2000 number of epochs and the second model also for 2000 epochs. In both the models, loss is calculated using Mean Squared Error (MSE) for every epoch and it is used as a factor to converge the model using early-stopping criteria. Adam is used as the optimization technique in the model [9]. Along with that, for analyzing the results, we calculated R Squared (R2) error and accuracy for each epoch. Since this a regression-based model, relative accuracy is used with 20% error margin. For validating the models, K-fold validation technique is used [7]. TensorFlow and Keras are used as the frameworks are used to create the neural network models [6] [8].

IV. ALTERNATIVE MODELS

One of the alternative methods we have done is using the support vector regression (SVR) method [7]. In this method, we created two SVR models using dataset A and dataset B. Kernel function used in both the model is sigmoid kernel function [7]. Also like in main model, we did validation using K-fold validation method and calculated accuracy using relative accuracy with error margin 20%. Scikit-learn library is used for making SVR model [7].

Other alternatives are with the different neural network architecture. We tried different models by varying nodes and layers in the backpropagation neural network and did validations using K-fold validation technique.

V. RESULTS

Matplotlib is used to plot the graphs [10]. Following are the result graphs obtained from both the datasets.

A. Dataset A

In case of dataset A Fig.2 and Fig.3 are the testing results using 14 samples of SVR and neural network models respectively. Red dot represents the predicted CPU load and blue dot represents the original CPU load. As in both the figures, prediction using neural network model is better than the support vector regression model. Fig.4, Fig.5 and Fig.6 are

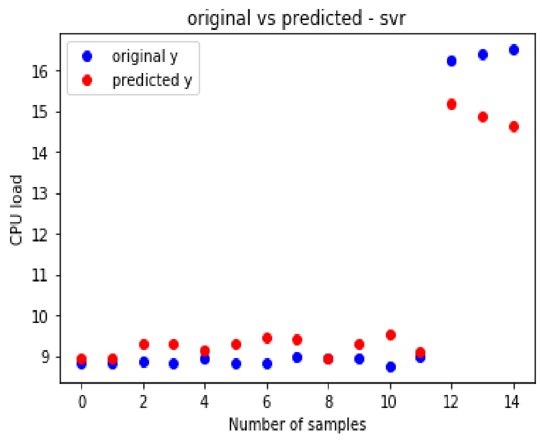

Fig. 2. Prediction result of test dataset A using SVR.

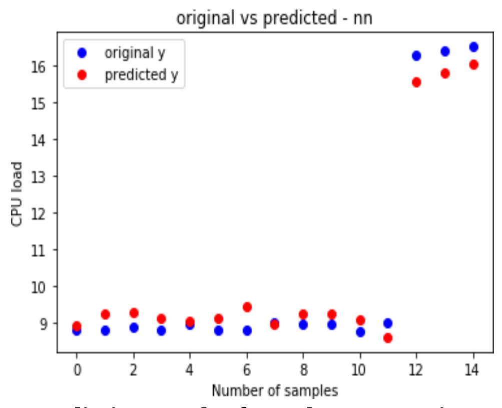

Fig. 3. Prediction result of test dataset A using neural network.

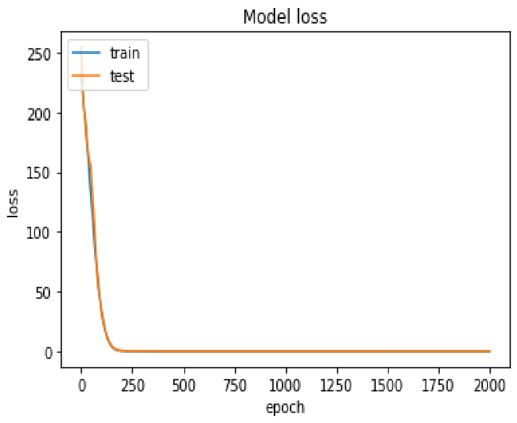

Fig. 4. Loss (MSE) graph of neural network model using dataset A.

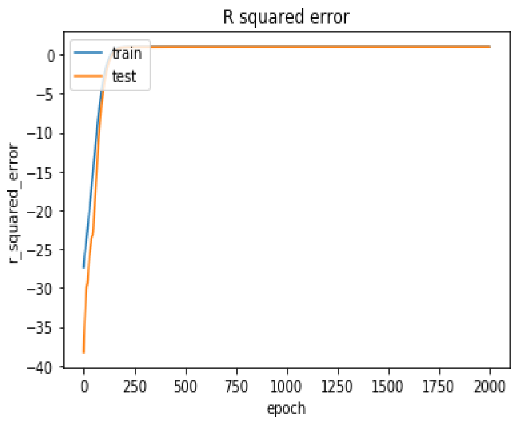

Fig. 5. R2 error graph of neural network model using dataset A.

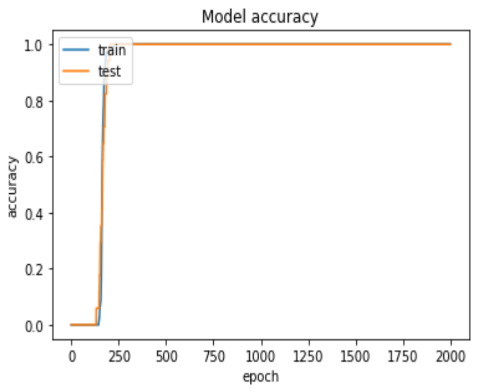

Fig. 6. Accuracy graph of neural network model using dataset A.

MSE, R Squared error and accuracy graphs during the epochs of the neural network model respectively. In these figures, the blue line represents the values obtained using the train dataset and orange line represents the values obtained using the validation dataset during the epoch. It is clearly shown in graph that the errors are reducing, and accuracy is increasing over each epoch.

In case of support vector regression model, during validation of the main model, MSE is 2.12, R Squared error is -0.057 and accuracy is 83.33% and during the prediction of test dataset, MSE is 0.604, R squared error is 0.932, and accuracy is 86.67%.

But, in case of neural network model, during validation of the main model, MSE is 0.025, R Squared error is 0.996, and accuracy is 100% and during the prediction of test dataset, MSE we got is 0.152. R squared error is 0.983 and accuracy is 93.33%.

B. Dataset B

Similarly, in case of dataset B, red dot represents the predicted CPU load and blue dot represents the original CPU load. Fig.7 and Fig.8 are the testing results using 14 samples using SVR and neural network models respectively. Predic- tion using neural network model is better than the support vector regression model. Fig.9, Fig.10 and Fig.11 are MSE, R Squared error and accuracy graphs during the epochs of the neural network model respectively. Here also, it is clearly shown in graph that the errors are reducing, and accuracy is increasing over each epoch.

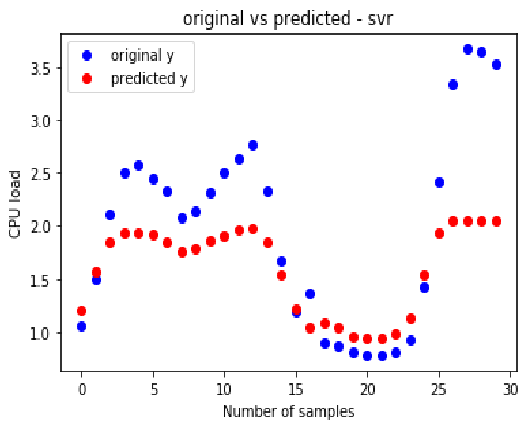

Fig. 7. Prediction result of test dataset B using SVR.

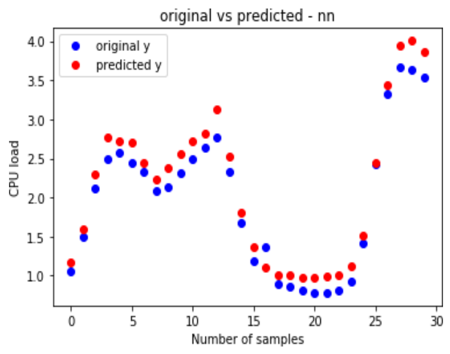

Fig. 8. Prediction result of test dataset B using neural network.

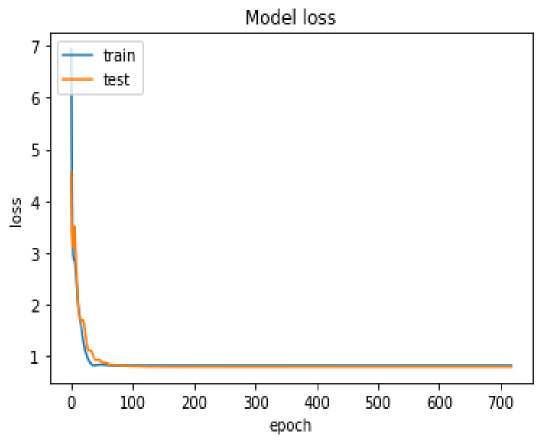

Fig. 9. Loss (MSE) graph of neural network model using dataset B.

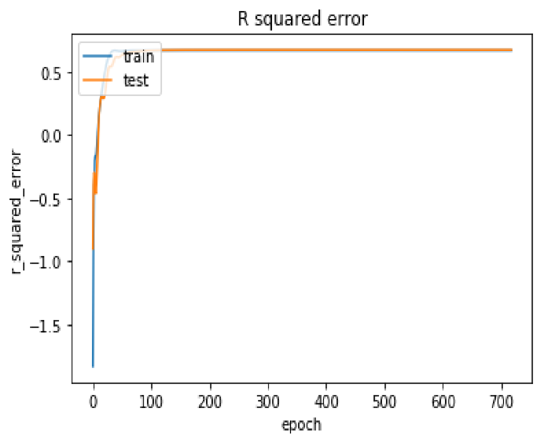

Fig. 10. R2 error graph of neural network model using dataset B.

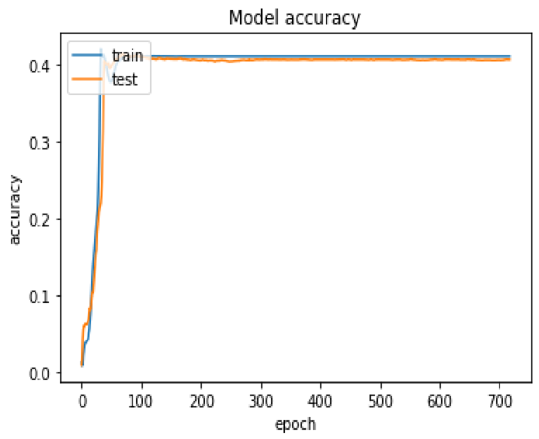

Fig. 11. Accuracy graph of neural network model using dataset B.

In case of support vector regression model, during validation of the main model, MSE is 1.93, R Squared error is -0.182, and accuracy is 33.99% and during the prediction of test dataset, MSE is 0.433, R squared error is 0.464, and accuracy is 40%. But, in case of neural network model, during validation of the main model, MSE is 0.791, R Squared error is 0.669, and accuracy is 40.62% and during the prediction of test dataset, MSE we got is 0.043. R squared error is 0.946 and accuracy is 83.33%.

VI. CONCLUSION

From the above results, it has been concluded that using traffic profile we can predict the CPU load. We used two kinds of datasets, one from the test report and another from real time. Using these datasets, we created two models for each,

SVR model and neural network model. From the results, we obtained that during prediction, neural network model gives better result than the support vector model. In case of dataset A, in the result graphs the neural network model is converging after 300 epochs, but due to minor decrease over each epoch in loss, early-stopping is not applying. Also, R2 error is a negative value during validation in SVR. This essentially means that the mean value is closer to the measured data points than the estimated ones. Some values are having different range which diverge much more than the other sample points, which is leading to this result. In case of dataset B the neural network model is converging at 750 epochs. Also, in case of dataset B some samples had very less value, which leads to makes the model to learn the base CPU load which is not dependent on the traffic profile. It has been affecting the prediction of some results as it is clearly shown in the prediction graphs. To improve the performance of the prediction on neural network model more datasets needed to be trained from different traffic profiles.

VII. ACKNOWLEDGEMENT

The authors would like to express our sincere gratitude to Timo J Holopainen and Qin Zhang from Nokia Solutions and Networks Pvt. Ltd. for their support, suggestions and valuable feedback during this research work.

REFERENCES

1. Boulmalf, M., Abrache, J., Aouam, T. and Harroud, H. Traffic Analysis for GSM Networks. IEEE, (978-1-4244-3806-8/09/), pp.498-503, 2009
2. Ct, D. Using Machine Learning in Communication Networks [Invited]. Journal of Optical Communications and Networking, 10(10), p.D100, 2018
3. Panda, M. and Prasad Padhy, S. Traffic Analysis and Optimization of GSM Network. IJCSI International Journal of Computer Science Issues, Special Issue, ICVCI, 1(1), pp.28-31, 2011
4. E.O, O., E.N, O. and M.A, A. Determination of Voice Traffic Busy Hour and Traffic Forecasting in Global System of Mobile Communication (GSM) in Nigeria. IEEE 11th Malaysia International Conference on Communications, (978-1-4799-1532-3/13/), pp.184-189, 2013.
5. Ricciato, F. Traffic monitoring and analysis for the optimization of a 3G network. IEEE Wireless Communications, 13(6), pp.42-49, 2006.
6. ”TensorFlow”, TensorFlow. https://www.tensorflow.org/. [Accessed: 13- Jun- 2019].
7. ”scikit-learn: machine learning in Python scikit-learn 0.21.2 documen- tation”, Scikit-learn.org. https://scikit-learn.org/stable/. [Accessed: 13- Jun- 2019].
8. ”Home - Keras Documentation”, Keras.io. https://keras.io/. [Accessed: 13- Jun- 2019].

9. ”An overview of gradient descent optimization algorithms”, Sebastian Ruder. http://ruder.io/optimizing-gradient-descent/. [Accessed: 13- Jun- 2019].
10. ”Matplotlib: Python plotting Matplotlib 3.1.0 documentation”, Mat- plotlib.org. https://matplotlib.org/. [Accessed: 13- Jun- 2019].
11. S. Angadi and R. Gopalapillai, ”Mining Network Stream Data for Self Learning Networks”, IEEE - 43488, 2018.
12. B. Mounika, G. Raghu and R. Jeyanthi, ”Neural Network based Data Validation Algorithm for Pressure Processes”, International Conference on Control, Instrumentation, Communication and Computational Tech- nologies (ICCICCT), pp. 1223-1227, 2014.
13. R. Parkinson, “Traffic engineering techniques in telecommunications”, Infotel Systems Corp.
14. L. Moglia, “Control Plane Performance Comparison in Virtualised Base Station Controller”, M.S. thesis, Tampere University of Technology, March 2015. [Online]. Available:https://dspace.cc.tut.fi/dpub/bitstream/handle/123456789/22863/Moglia.pdf?sequence=3&isAllowed=y
15. Venkatesan, C., P. Karthigaikumar, Anand Paul, S. Satheeskumaran, and R. Kumar. "ECG signal preprocessing and SVM classifier-based abnormality detection in remote healthcare applications." IEEE Access 6 (2018): 9767-9773..