What Machine Learning Predictor Performs Best for Mobility Prediction in Cellular Networks?

Hana Gebrie, Hasan Farooq and Ali Imran
University of Oklahoma, Tulsa, USA 74135
Email: Hgebrie@oru.edu,{hasan.farooq, ali.imran}@ou.edu

***Abstract*—It is envisaged that the future cellular networks (5G) will be able to meet the promising capacity and quality of experience requirements through extreme network densification and conglomeration of diverse technologies. It is easy to fathom that efficient management of such a convoluted network will be one of the big challenges faced by 5G. To cope with this challenge, Self-Organizing Networks (SONs) that were originally designed for legacy networks with reactive approach needs to be transformed to proactive paradigm. This radical transformation is possible only if the future network state can be predicted beforehand by harnessing historical network data. Mobility prediction is one of the key enablers of Proactive SON which enables efficient resource management. In this paper, we perform comparative analysis of four mobility predictors: Deep Neural Network (DNN), Extreme Gradient Boosting Trees (XGBoost), Semi-Markov, and Support Vector Machine (SVM). Our investigation is based on realistic synthetic dataset of eighty-four mobile users generated through realistic Self-similar Least Action Walk (SLAW) mobility model. We evaluate the effectiveness of each model not only based on the model's ability to predict the future location of mobile users but also the time each algorithm takes to be fully trained and perform such prediction. XGBoost stands out as clear winner among all predictors considered with high accuracy of 90%. Its high prediction accuracy enables high energy saving gain of above 80% when it is employed for driving proactive energy saving SON solution.**

***Index Terms*—Deep Neural Networks, mobility prediction, self-organizing networks (SON), semi-Markov, SVM, XGBoost, 5G.**

I. Introduction

Mobile communication has become one of the fastest growing segments in the communications industry. In the past decade, the cellular industry has witnessed an exponential growth of mobile traffic that is anticipated to further grow about ten-thousand times compared to 2017 figures. Moreover, number of devices connecting to mobile networks is expected to reach 125 billion by 2030. This trend is resulting in an unprecedented demand for infinite capacity with zero latency (Quality of Experience [QoE]). Fifth-Generation (5G) cellular network is presumed to meet aforementioned ambitious requirements. Network densification, diversity in node types, decoupled control/data planes, network virtualization, infrastructure sharing, concurrent operation at multiple frequency bands, and simultaneous use of different medium access are considered to be important flavours of envisioned 5G networks [1]. However, increased complexity is the price to pay for the promising gain yielded by 5G. This high complexity stemming mainly from ultra-densification will aggravate number of issues like higher CAPEX/OPEX, energy consumption, seamless mobility management to name a few [1], [2].

Proactive Self-Organizing Networks (SONs) are considered to be panacea of this complexity crunch [3] that allows operators an unprecedented opportunity to optimize their network performance in real time and extract the benefits that 5G offers. Machine learning (ML) empowered mobility prediction is the cornerstone of this proactive network optimization and zero-touch automation paradigm which identifies future target base stations of users based on their mobility history (see Fig. 1). An accurate user mobility prediction in mobile networks provides efficient resource and handover management, which can avoid unacceptable degradation of the perceived quality [3].

Despite apparent randomness in individual trajectory, several studies based on real datasets collected from mobile networks have shown deep rooted regularity and sufficient predictability in user trajectories [2], [4]. Several studies exist that exploit various approaches for mobility prediction— refer to comprehensive surveys in [3], [5]. Markov chain-based predictor has been the common choice for almost all of the mobility prediction studies due to its attractive small space/time complexity [5]. However, with the computational resources present today, machine learning predictors can be viable alternative. Hence there is need to explore performance of machine learning predictors for mobility prediction in cellular networks. In this paper, we perform comparative performance analysis of three relatively powerful machine learning algorithms (i) deep neural networks (DNN) [6], (ii) extreme gradient boosting trees (XGBoost) [7] and (iii) support vector machine (SVM) [6] that got their classification capability acknowledged by the computer vision community. We also benchmark their performance against semi-Markov-based mobility prediction model [2].

The real challenge here was selection of a mobility trace generation model that realistically represents behavior of actual cellular network users. Several such models have been proposed recently in literature such as SLAW, SMOOTH, Truncated Levy Walk etc., [8]. Based on an extensive analysis of these models, we chose SLAW (Self-similar Least Action Walk) [9] mobility model. Unlike commomly used random walk models where movement at each instant is completely random, SLAW has been shown to be a highly realistic mobility model. It exhibits all the characteristics of real world human mobility, i.e., (i) truncated power-law flights and pause-times: the lengths of human flights which are defined to be

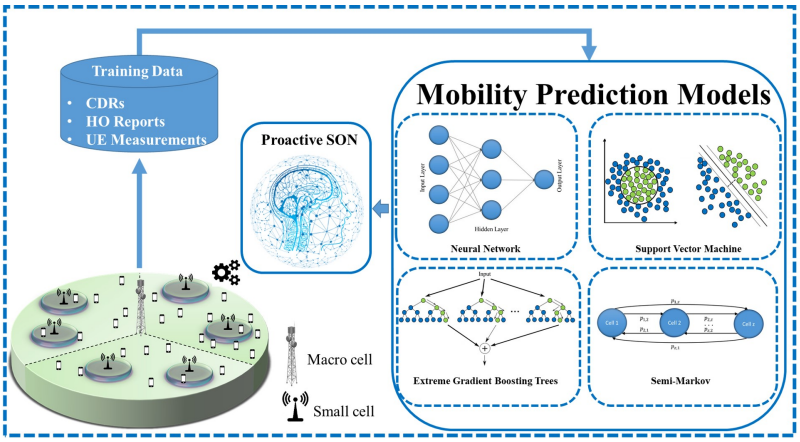

Fig. 1. Mobility Prediction empowered Proactive SON Paradigm

straight line trips without directional change or pause have a truncated power-law distribution (ii) heterogeneously bounded mobility areas: people mostly move only within their own confined areas of mobility and different people may have widely different mobility areas (iii) truncated power-law inter- contact times: the times elapsed between two successive contacts of the same persons follows truncated power law distribution and (iv) fractal waypoints: people are always more attracted to more popular places. Therefore, the accuracy of machine learning predictors tested using mobility traces generated by SLAW is very likely to represent their true performance in real network. In this work, predictors trace an individual user's patterns regarding the connected base station rather than capturing multiple users' trajectory in general as proposed in studies like [10]. The size of dataset for an aggregated mobility prediction model will be significantly large, so it would potentially yield a consistent accuracy rate by law of large numbers. However, that does not necessarily mean it will give a higher accuracy rate. Since the number of base stations visited by users varies from one another, a user prediction model would possibly have relatively higher randomness in their mobility patterns which causes lower predictability rate. In a hexagonal planned network, one base station can have a maximum of six neighboring cells, which means there would be seven possible future locations associated with a person's current position—one itself and remaining the six neighbors. The user moblity prediction model will potentially predict one of the seven possibilities (classes or cells) at each timestep.

The contributions of this paper can be summarized as follows:

1) We employ 3GPP compliant simulations and leverage SLAW model for generating mobility traces that are used for training/testing predictors. Therefore comparison performed based on SLAW generated traces is likely to represent predictors true performance in actual network mobility traces.
2) We perform comparative analysis of three commonly used predictors in machine learning (deep neural networks, extreme gradient boosting trees, support vector machine) and semi-Markov model. We compare performance of these predictors in terms of prediction accuracy and time complexity. Results indicate xgboost outperfoms all others in terms of prediction accuracy while semi-Markov-based model performs best in terms of time complexity. Another insight provided by this paper is positive effect of number of previous locations visited by UE on predictor's performance.
3) As a case study, we analyze the effect of predictors accuracy on performance of proactive energy saving (ES) SON solution. We observe that prediction accuracy has profound effect on gain yielded by proactive ES solution with XGBoost yielding 80.68% energy reduction gain in whole network as compared to current industrial practice of Always ON strategy.

II. Machine Learning Predictors for Mobility Prediction

Deep learning has been producing excellent outcomes in the filed of machine learning, specifically, for image and audio recognition. Deep learning extends the overall field of Artificial Intelligence (AI) by showing its distinct advantages in large datasets. Due to the advancement in field of deep learning, it is common notion that other algorithms like SVM and XGBoost would no longer be efficient for prediction problems. However, that is not the case as no algorithm rules over others in all cases. The benefits of deep learning manifest more in a complex featured engineered and very massive set of data. Also, its advantages manifest more in images and audios as stated previously. Deep learning can also be used on tabular datasets and still can perform well, but it can be surpassed by other methodologies, like XGBoost. Accordingly, different algorithms can suit well for various problems, and our contribution is to qualify performance of each algorithm to find the best solution for mobility prediction in mobile networks.

A. Mobility Prediction Using Deep Neural Network

Among the various deep learning models, a feed-forward DNN is used to train the model and evaluate its performance. DNN uses a cascade of multiple layers of non-linear processing units. It is composed of an input layer, output layer and number of hidden layers. In the case of feed-forward networks,

the dataset flow from the input layer to the output layer without looping back. The outputs of one layer serve as the inputs for the next layer. Each layer is fully connected to one another, the input layer with the first hidden layer, the first hidden layer with the second, and so on up to the output layer. Our model uses the Rectified Linear Unit (ReLU) activation function on the input and hidden layers. ReLU has become very popular in the last few years. It improves neural networks by rapidly accelerating the convergence of stochastic gradient descent compared to the *sigmoid/tanh* functions. Mathematically it is defined as:

$$f(x) = W \max(0, x) \tag{1}$$

where x is the input data, $f(x)$ is the activation, and the function $\max(0, x)$ is a non-linearity that is applied element-wise. A three-layer neural network could analytically look like:

$$f(x) = W_3 \max(0, W_2 \max(0, W_1 x)) \tag{2}$$

where all of W_3,W_2,and W_1 are parameters to be learned. The primary benefit of using ReLUs is having a reduced likelihood of vanishing gradient. The gradient of the ReLu function is either 0 for $x < 0$ or 1 for $x > 0$. Since multiplying the gradients will neither vanish nor explode, so we can potentially use as many hidden layers as possible. However, the zero gradient on the left-hand side has its own problem, known as dead-neurons which cause the output to be always zero. We applied a *softmax* activation function at the output layer to obtain a distribution over the K classes. It is common practice in classification problems to use *softmax* as a classifier at the end of the neural networks. It "squashes" a K-dimensional vector z of arbitrary real values to K-dimensional vector $\sigma(z)$ of real values, where each entry is in the range (0, 1] and the sum of the components of the output vector is equal to one—in other words, it turns numbers into probabilities. The *softmax* function is given by:

$$\textit{softmax}(\text{z}) = \sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for } j = 1, ..., K \tag{3}$$

When the last layer is a dense layer with *softmax* activation, we will be using the following formula to calculate the loss L, which is the summation of the errors made for each example in the training and validation set:

$$L = -y \times \log(\hat{y}) \tag{4}$$

The target variable is a matrix of one and zeros indicating which class the corresponding input belongs. The ground truth y gives all the probability to the class number one and leaves the zero values. Therefore, to calculate the loss, it only uses the matching term from the estimate $\hat{y}$ e.g., suppose the true label y is [1 0 0 0 0] and the predicted $\hat{y}$ is [0.2 0.3 0.1 0.3 0.1] then $L = -(1 \times \log(0.2) + 0 \times \log(0.3) + ...) = -\log(0.2) \approx 0.699$. Unlike the loss function, the accuracy rate is computed in percentage. The accuracy rate in (5) is defined as ratio of total number of true positives (TP) and true negatives (TN) to total population. While the TP shows the correctly identified, the false positive (FP) is the incorrectly identified outputs. On the other hand, TN indicates correctly rejected and the false negative (FN) shows the incorrectly rejected.

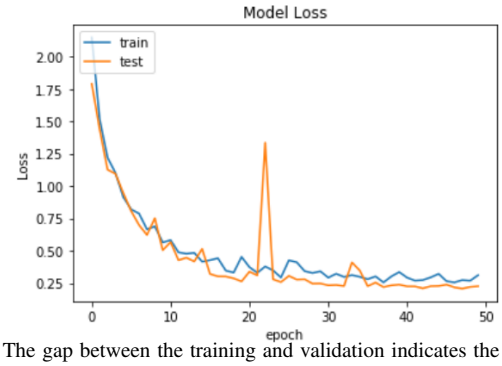

Fig. 2. The gap between the training and validation indicates the amount of overfitting in loss.

$$Accuracy = \frac{\sum TP + TN}{TP + TN + FP + FN} \tag{5}$$

We used repeated k-fold cross-validation with $k = 4$ and $n = 2$ to split the training data into k equal partitions which are known as folds and repeat it n times. The two most important parameters in DNN are depth and width of the model, which also known as layers and number of neurons respectively. After trying various combinations, the final model chosen had a total of six hidden layers with sixty neurons in each layer. Having many hidden units within a layer increases the accuracy rate albeit at increased risk of overfitting. Unlike the hidden layers, the output layer has the number of neurons corresponding to the number of target classes. The other major parameters that are highly related to the running time and accuracy rate are batch size and number of epochs. While the batch size defines the number of patterns to be read at a time and kept in memory, the epochs designate the number of times the network analyzes the entire dataset during training. Our model gave best results with ten batches and fifty epochs. The performance of the model is evaluated at each epoch. Finally, we get four outputs: accuracy rate, cross entropy loss, validation accuracy, and validation loss.

The result in Fig. 2 show loss rate, which is useful to track overfitting in the model. If the gap increases as the number of epochs increases, then the model is overfitted and needs tuning of hypermeters and network topology adjustment. According to the diagram, our model is not overfitted.

B. Mobility Prediction Using Extreme Gradient Boosting

We evaluated another impressive ensemble of tree method called XGBoost for human mobility prediction in cellular networks. XGBoost is a scalable variant of Gradient Boosting Machine (GBM). The tree ensemble is made out of a set of classification and regression trees (CART). The trees grow one after another in order to reduce the misclassification rate. The most significant factor behind the success of XGBoost is its ease of use, ease of parallelization, and incredible predictive accuracy [11]. Mathematically, model is given as [7]:

$$\hat{y}_i = \sum_{k=1}^{K} f_k x_i, f_k \in F \tag{6}$$

where K is number of trees and F is set of all possible CARTs. The regularized objective is given by:

$$L(\theta) = \sum_{i}^{n} l(\hat{y}_i, y_i) + \sum_{k}^{K} \Omega(f_k) \tag{7}$$

where $\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$ and n is number of samples. The term l indicates the differentiable loss function that measures the difference between the target y_i and the prediction $\hat{y}_i$. w is the vector of scores on leaves, Ω is the other regularization term that penalizes the complexity of the model in order to prevent overfitting and T is total number of leaves. λ and γ are constant coefficients that control the degree of regularization. The prediction $\hat{y}_i$ at step t is expressed as $\hat{y}_i^{(t)}$. Since the model is trained in an additive manner, therefore, we will need to add f_t in order to be able to optimize. At time step t:

$$L(\theta)^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}^{(t-1)} + f_t(x_i)) + \Omega(f_t) \tag{8}$$

where $\hat{y}^{(t)} = \hat{y}^{(t-1)} + f_t(x_i)$. Taylor expansion of the loss function to the second order can be used to quickly optimize. Then, the simplified objective function can be written as follows at step t:

$$L(\theta)^{(t)} = \sum_{i=1}^{n} [g_i f_t(x_i + \frac{1}{2} h_i f_t^2(x_i)] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{9}$$

where $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$, $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ and $I_j = \{i | q(x_i) = j\}$ is the set of indices of data points assigned to the j^{th} leaf. The regularized objective with the t^{th} tree can be re-written as follows:

$$L(\theta)^{(t)} = \sum_{j=1}^{T} [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2}(\sum_{i \in I_j} h_i + \lambda) w_j^2] + \gamma T \tag{10}$$

Using the following formula, we can calculate the instance set of the leaf j and the optimal leaf weight w_j^*.

$$w_j^* = \frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{11}$$

and the corresponding optimal value will be

$$L(\theta)^{(t)^*} = -\frac{1}{2} \sum_{j=1}^{T} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T \tag{12}$$

We can use (12) as a scoring tool to compute the quality of a tree structure $q(x_i)$. However, it is preferable to use an algorithm that starts with a single leaf then adds branches to the tree. Then, we can compute the loss reduction after the split as follows:

$$L_{split} = \frac{1}{2}[\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda}] - \gamma \tag{13}$$

where I_L is an instance of the left node, I_R is an instance of the right node, and I is the summation of the two instances of nodes. In order to get a better structure score, we need to sum up the gradient and the second order gradient statistic on each leaf and apply (13). The smaller the score, the better the structure. In our study, XGBoost model obtained best performance when the minimum child weight, maximum depth, column sample, step size and shrinkage η that controls the learning rate and overfitting was set to 5, 3, 0.8, 50, and 0.01 respectively. Other tree booster parameters were set to their default values.

C. Mobility Prediction Using Support Vector Machine

The third prediction mechanism we used is the SVM model also known as large margin classifier. SVM classifier maps a set of inputs in a higher dimensional feature space through some linear and non-linear mapping to increase the distance between different classes [6]. The idea behind SVM is finding a decision boundary between two classes and construct a hyper plane that has the largest distance to the nearest training sample of either class. We used an instance of non-linear SVM with radial basis function (RBF) kernel. The RBF kernel on the two sample x_1 and x_2, is represented as feature vectors in some input space and the kernel can be calculated as:

$$K(x_1, x_2) = \exp(-\frac{||x_1 - x_2||^2}{2\sigma^2}) \tag{14}$$

where K implies the kernel, σ is an RBF parameter. Let $\gamma = \frac{1}{2\sigma^2}$ then (14) can be re-written as:

$$K(x_1, x_2) = \exp(-\gamma ||x_1 - x_2||^2) \tag{15}$$

The new RBF parameter γ and the SVM regularization parameter C are optimized on a subset of the training dataset based on a grid search. Parameters that have a higher impact on SVM model performance are K, C, and γ where the kernel specifies the algorithms kernel type, C is the SVM regularization parameter, and γ designates a kernel coefficient. In our study, SVM model uses RBF kernel which is useful for non-linear hyperplane, C uses the default value 1, and γ also set to the default value because the model performs best with such a parameter set up in our study. Parameters like degree, shrinkage, probability, tolerance for stopping criterion, verbose, and max iteration were set to their default values. The last parameter "cache_size", which specifies the size of the kernel cache, was set to 200.

D. Mobility Prediction Using Semi-Markov

The last approach we use to predict future location of a mobile user is the semi-Markov prediction model since they have been largely used by the published studies [5]. The Markov models use probabilistic reasoning to predict the future state of a mobile user. Markov condition assumes that future state prediction only depends on the current state of the user and thus is independent of all previous memory [3]. For brevity, we didn't include theory of semi-Markov-based model here. Readers interested in the applications and specifications

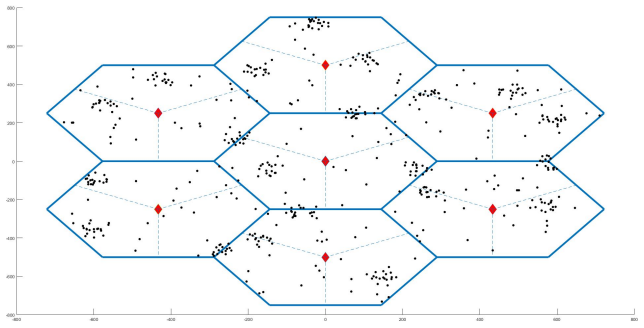

Fig. 3. Network Topology with black dots denoting UEs and red diamonds the macro base stations.

of the model along with detail mathematical derivations, please refer to [2].

III. Dataset Creation

We created synthetic dataset of 84 mobile UEs by employing 3GPP compliant LTE simulator in MATLAB. Network topology consisted of 7 macro cells, each macro comprising of three sectors thus total of 21 cells (Fig. 3). Mobility patterns were generated for one week with one-minute granularity through realistic SLAW mobility model. Each user had a total of 10080 observations. Approximately 85% of the data was used for the training set, and the remaining 15% used to test the prediction accuracy. The training dataset was also split into two for training (75% of 85%) and validation set (25% of 85%) by using four-fold cross-validation. The cross-validation uses one-fold for the testing set and the union of the rest of the folds for the training set. For input features, we first use the current location of the mobile user and the sojourn time which is a total time a person spends in one cell, with all four predictors. Then for DNN we considered additional features corresponding to three previous locations.

IV. Results and Analysis

One of the primary reasons to evaluate machine learning models is to be able to choose the best available model and estimate how well a given model is likely to perform in a real network. We evaluated the effectiveness of each model not only based on the model's ability to predict the next location of mobile users but also the time each algorithm takes to be fully trained and perform such prediction. Deploying an algorithm that has the highest accuracy is crucial to assure a great operability of SON functions [3] but not sufficient. Functions such as mobility management and HO optimization are highly dependent on time. Not only they rely on the training time but also the time an algorithm takes to predict the desired SON function. According to the experimental results, the XGBoost model produced the best accuracy rate (see Fig. 1). The figure shows the average prediction and training accuracy rate of eighty-four mobile users using all the four algorithms. The blue bar indicates the percentage of correctly classified during training time while the green bar designate the percentage of correctly classified during testing the model with a dataset that has not been previously seen by the network. The SVM model

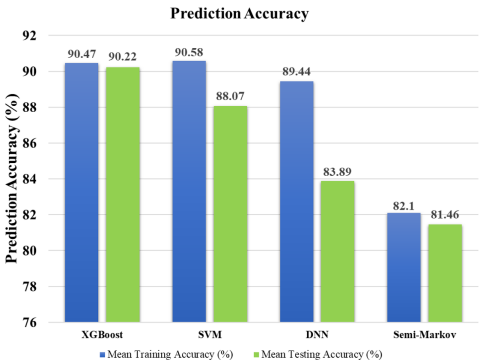

Fig. 4. Prediction Accuracy of the Predictors

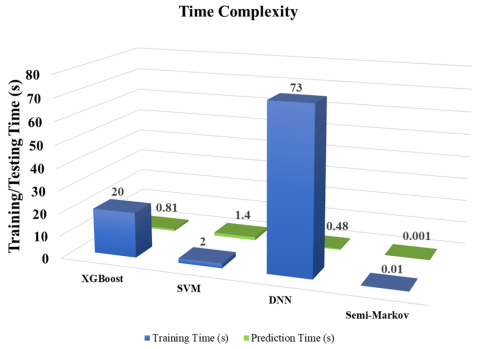

Fig. 5. Execution time of the Predictors

gave a relatively high accuracy rate for training dataset but not surpassed the XGBoost's outcome for testing dataset. The performance of the DNN model also achieved a satisfactory result compared to the semi-Markov, however its training time is quite higher than all the other models (see Fig. 5). This is because training DNN with mini-batch based stochastic gradient descent requires frequent serial training and scanning the whole training data set many passes before reaching the asymptotic region. On the other hand, the semi-Markov-based model gave a relatively smaller accuracy rate, but the model executes within a very small execution time, which is one of the important factors of mobility prediction evaluation techniques.

Next, we analyzed the effect of number of previous locations

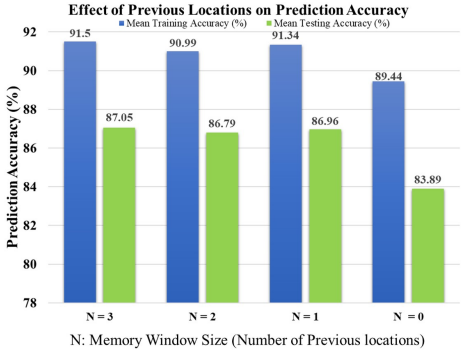

Fig. 6. Training and Testing Accuracy rate of the DNN with increase in memory window size

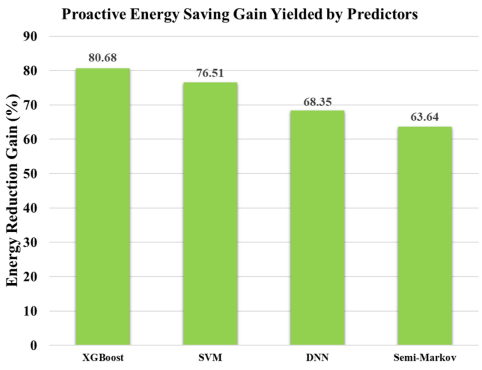

Fig. 7. Energy Reduction Gain

on the predictors performance. Here we selected DNN as representative predictor from ML family. We trained and tested DNN by considering the current location of the user, sojourn time and then added previous locations. Our experimental results (Fig. 6) show that the more input dimension the model has, the more consistent and accurate the output becomes. It was observed that the models' execution time is not affected by the number of previous locations used in the DNN.

V. Utility of Mobility Prediction in Proactive SON

The presented spatiotemporal mobility predictors can empower SON functions like energy saving (ES) [12] and transform them from reactive to proactive. The AURORA framework presented by us in [12] uses mobility prediction to determine future cell loads that in turn is then used to proactively schedule small cell sleep cycles (For details, refer to [12]). Based on the intelligence gained from the mobility prediction i.e., a proactive energy saving optimization problem is formulated to minimize the energy consumption by switching OFF under-utilized small cells. We gauged performance of AURORA framework using DNN, SVM, XGBoost and semi-Markov as mobility prediction models leveraging information in Fig. 13 of [12]. The average Energy Reduction Gain (ERG) computed in (16) [12] of proactive ES leveraging these mobility predictors against scenario for always ON strategy is plotted in Fig. 7.

$$ERG = \left(\frac{EC_{alwaysON} - EC_{proactiveES}}{EC_{alwaysON}}\right) \times 100\% \quad (16)$$

In (16), EC_x is energy consumption (Joules/bit) of scheme x. It is observed that as expected the gain of proactive energy scheme increases with the prediction accuracy and is found maximum for XGBoost. This proactiveness, enabled by mobility prediction, make it possible for cellular networks to meet 5G ambitious latency and QoS requirements.

VI. Conclusions

In this paper, we examine the performance of four powerful predictors on the prediction of human mobility patterns which is essential to overcome mobile traffic, manage mobility, and reduce the consumption of energy in future cellular networks. Experimental results show that XGBoost is indeed an effective mobility prediction algorithm. It produces an impressive predictive accuracy (90.22%), as well as considerably lower execution time. The SVM also gives a relatively high accuracy rate compared to the DNN and semi-Markov. The performance of the DNN model also achieves a significantly high result; it produces above 80% accuracy rate. Yet, the training time of the DNN algorithm is higher than all the other models and also the output results of the DNN change at some degree as we re-train the model which makes the prediction inconsistent. However, as we increase the number of features (previous locations), we obtain a more consistent and accurate output. On the other hand, the semi-Markov model gave a relatively smaller accuracy rate, but the model executes within a minimal execution time which is one of the essential factors in meeting stringent QoE requirements of 5G and beyond networks.

For future works, we will analyze robustness of these predictors against variations in training dataset sizes. Also, we will benchmark performances based on mobility traces gathered from real network CDRs.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant Numbers 1619346, 1559483, 1718956 and 1730650. The statements made herein are solely the responsibility of the authors.

References

[1] A. Imran, A. Zoha, and A. Abu-Dayya, "Challenges in 5g: how to empower son with big data for enabling 5g," *IEEE Network*, vol. 28, no. 6, pp. 27–33, Nov 2014.

[2] H. Farooq and A. Imran, "Spatiotemporal mobility prediction in proactive self-organizing cellular networks," *IEEE Communications Letters*, vol. 21, no. 2, pp. 370–373, Feb 2017.

[3] P. V. Klaine, M. A. Imran, O. Onireti, and R. D. Souza, "A survey of machine learning techniques applied to self-organizing cellular networks," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2392–2431, Fourthquarter 2017.

[4] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[5] H. Zhang and L. Dai, "Mobility prediction: A survey on state-of-the-art schemes and future applications," *IEEE Access*, vol. 7, pp. 802–822, 2019.

[6] K. P. Murphy, *Machine learning : a probabilistic perspective*. Cambridge, Mass. [u.a.]: MIT Press, 2013.

[7] X. Developers, "Xgboost documentation," 2016. [Online]. Available: https://xgboost.readthedocs.io/en/latest/

[8] M. Gorawski and K. Grochla, *Review of Mobility Models for Performance Evaluation of Wireless Networks*. Cham: Springer International Publishing, 2014, pp. 567–577.

[9] K. Lee, S. Hong, S. J. Kim, I. Rhee, and S. Chong, "SLAW: A New Mobility Model for Human Walks," in *IEEE INFOCOM 2009 - The 28th Conference on Computer Communications*. IEEE, apr 2009, pp. 855–863.

[10] J. Capka and R. Boutaba, "Mobility prediction in wireless networks using neural networks," in *Management of Multimedia Networks and Services*, J. Vicente and D. Hutchison, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 320–333.

[11] I. B. Mustapha and F. Saeed, "Bioactive Molecule Prediction Using Extreme Gradient Boosting," *Molecules*, vol. 21, no. 8, 2016.

[12] H. Farooq, A. Asghar, and A. Imran, "Mobility Prediction-Based Autonomous Proactive Energy Saving (AURORA) Framework for Emerging Ultra-Dense Networks," *IEEE Transactions on Green Communications and Networking*, vol. 2, no. 4, pp. 958–971, Dec 2018.