RNN-Based User Trajectory Prediction Using a Preprocessed Dataset

Nasrin Bahra and Samuel Pierre
Mobile Computing and Networking Research Laboratory (LARIM)
Department of Computer and Software Engineering
Polytechnique de Montréal
Montreal, Canada
nasrin.bahra@polymtl.ca, samuel.pierre@polymtl.ca

***Abstract*—Future mobile networks are rightly expected to face the prospect of limited available resources. Continuous technological advances and growing number of mobile devices highlight the importance of further improving the performance of mobile networks. User mobility poses technical problems in network management. It is essential to ensure a satisfactory level of quality of service for users. To achieve this goal, self organizing networks (SONs) are potential solutions to fulfill the requirements of users using learning algorithms. In this paper, we propose an intelligent mobility model to predict future trajectory of the mobile user in mobile networks. The proposed approach has two main parts, including mobility data preparation and user mobility prediction. Our primary focus is on providing a carefully tailored mobility data from raw mobility datasets using line simplification techniques. Next, we use the accurately prepared data for learning user mobility behaviour and predicting user future trajectory using recurrent neural networks and its variants. Simulation results show a substantial decrease in execution time from 4616s to 932s for the best case. The proposed learning approach obtains a loss value of 0.10 using a model based on long short term memory (LSTM).**

***Index Terms*—trajectory prediction, neural networks, mobile networks, data reduction**

I. INTRODUCTION

Rapid technological advances in many areas, such as the wildly popular paradigm of internet of things (IoT), can provide plenty of golden opportunities for users in the mobile networks. However, user mobility can cause a range of problems such as increase in call dropping, increase in number of handovers, limited available bandwidth, connectivity issues and in general quality of service degradation. Mobile users are entitled to expect a reasonable level of quality of service (QoS) in the network. Therefore, it is crucial to further improve the network performance and provide high level services to meet the demands of the users in the mobile networks.

User mobility prediction as a part of intelligent mobility management is a promising solution to address the mentioned issues. This concept is defined in the context of self-organizing networks (SONs). These networks tend to eliminate the manual efforts in configuration, optimization and healing in the network management. Their focus is on developing fully automated functions for different components and tasks in the network. To achieve this objective, learning algorithms can be fully exploited to automatize the network management. Machine learning and deep learning algorithms can learn from the data and optimize the performance based on the past experiences [1].

From another stand point, there is a huge growth in the number of mobile devices in recent years. It is anticipated that there will be 5.7 billion mobile users by the year 2023 [2]. This rising trend in mobile users with mobile devices leads to an inevitable generation of mobility datasets. This huge amount of data can be used to broaden our understanding of users' needs and users' mobility behaviour. We can extract mobility patterns for predicting future trajectory of the user based on user's movement history. It is worth noting that the generated mobility data is a raw data that may contain noise and redundancy to some levels. It is highly recommended to apply some preprocessing techniques to the raw data and prepare the data before using it [3].

A large number of research areas can gain maximum benefit from predicting user mobility trajectory in the network. When network has the information of the user future locations, it can exploit this knowledge to better provide services in advance. It can be employed in passive resource allocation, handover management, passive bandwidth reservation, call admission control and optimized routing. As an example, for a mobility-aware handover management, network can complete handover preparation steps in advance. When handover is triggered, the procedure starts from the execution phase. This can significantly reduce the handover signalling overhead and latency [4].

In this paper, we propose an approach for user trajectory prediction. The proposed approach has two essential parts. First, we introduce an initial data preprocessing step to provide an appropriate dataset as an input for the model. This step is focusing on removing irrelevant data from the dataset and keeping only determining data points. We want to provide an analysis of different data reduction techniques. Second, we feed the prepared data to the mobility model to predict the future trajectory of the user. This model is based on recurrent neural network (RNN), long short term memory (LSTM) and gated recurrent unit (GRU). In the following, a part of recent related works are provided in Section II. Section III presents the proposed approach. Finally, simulation results and conclusion are provided in Sections IV and V, respectively.

II. RELATED WORKS

A wide range of works in the literature have studied user mobility prediction in mobile networks from different aspects [1], [3], [5]–[7]. From one point of view, we can divide these works into two popular categories: Markov-based approaches and learning-based schemes.

A large part of existing works relies on Markov models. In [8], authors proposed a trajectory prediction method based on hidden Markov models. They considered the impact of changing speed and introduced a self-adaptive algorithm for selecting parameters. Also, in [9], authors proposed an approach to predict the destination and mobility path using a second-order Markov model. However, it is shown that Markov models fail to effectively capture the long-term data dependency in the user past trajectories [10].

Moreover, there are numerous works based on learning algorithms in the literature for user mobility prediction. In [11], authors used deep learning algorithms such as neural networks, principal component analysis (PCA) and GRU for providing an accurate prediction for mobile IoT devices. An LSTM-based prediction approach was proposed in [12] to predict user's mobility based on mobility patterns. They also introduced a new parameter to control data transfer during the training stage. Another LSTM-based model was proposed in [13] for vehicle trajectory prediction. This method can make prediction for one to five seconds ahead. In [14], authors proposed an RNN-based model to predict the next location of the user. They introduced time/space-specific transition matrices to provide a novel spatial temporal prediction. Authors in [7] discussed location prediction in vehicular ad hoc networks (VANETs). They mentioned neural networks and machine learning algorithms as possible techniques to predict the future location of the vehicle. They concluded that artificial neural networks (ANNs) do not perform very well in terms of time complexity. Furthermore, they fail to learn the long-term data correlation between past movement of the user and accordingly result in poor accuracy performance.

Therefore, we need to introduce a model that can take advantage of neural networks and address the time complexity issue of them. We need a mobility model that considers the data correlation in the user trajectory to improve the accuracy performance. Moreover, we can deploy new data preprocessing techniques to effectively reduce the time complexity of the mobility prediction model.

III. PROPOSED APPROACH

In this section, we present our proposed approach. We propose to deploy line simplification methods to reduce the irrelevant data from a big dataset and keep only the appropriate data for the mobility learning model. Next, we use an RNN-based model to find the repetitive patterns in the data and predict the future trajectory of the user. Details regarding these two steps are provided in the following.

A. Data reduction

The primary step of the mobility learning is choosing an appropriate type of mobility data as movement history of the user. There are several types of mobility datasets such as GPS, Wi-Fi, cellular and location-based social networks datasets. We chose GPS data since it has the highest location accuracy. GPS dataset contains user location information in the form of geographical coordinates (i.e., longitude and latitude). However, raw GPS dataset is composed of user location information at almost each second with a high sample rate. All of this data is not necessarily needed for our purpose. Removing the irrelevant data results in almost the same performance accuracy but with much lower execution time. Fig. 1 depicts a mobile user GPS trajectory. The black line represents the actual path of the user and the raw information of user trajectory. Red points on the line are the points that we keep from the dataset and remove the rest. In this way, we have a similar trajectory with much fewer number of points.

There are several techniques to eliminate the unnecessary data. Here, we propose to apply line simplification methods to the raw trajectories of users. The core concept of line simplification approaches is to keep only the part of data that is essential to form the trajectory and delete the rest of the data points between them. Here, we provide a comprehensive analysis regarding some of the preprocessing techniques for data reduction and evaluate the impact of them on mobility learning.

Objective: Given the trajectory of user b with the length of n $[C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow ... \rightarrow C_n]$, we want to convert it to a simplified trajectory with the length of m $[C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow ... \rightarrow C_m]$, where $m < n$.

In the following, definitions of five line simplification methods are provided. It is worth mentioning that we hereafter refer to a geographical coordinate of GPS data as a location in this paper.

- $nth\ Point$: It is a naive algorithm that is relatively simple and fast. It does not involve forming any mathematical dependencies with the future or past locations in the user trajectory. Given a predefined number of consecutive locations (i.e., user trajectory), this algorithm keeps only the nth location point plus one random point in the set and removes the rest.
- $Reumann{-}Witkam$ $(R{-}W)$: Unlike nth point method, this algorithm considers the correlation of consecutive data points locally. It divides the trajectory into some sections and investigates neighbour locations based on a given threshold. In each section, it forms a line with the first two points in the trajectory. All the points in the section that are closer to the line more than a predefined threshold are deleted from the trajectory. This process is iteratively done for all the sections in the dataset.
- $Lang$: This method divides the trajectory into several segments including neighbour locations and considers a search region for each segment. For each point, if the perpendicular distance of its neighbour point is more

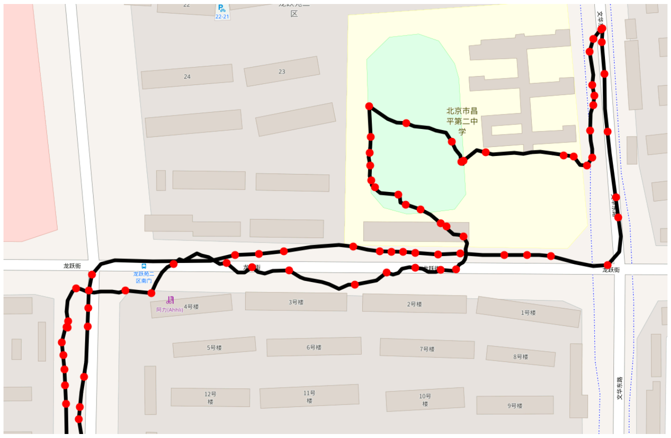


Fig. 1. User raw GPS trajectory. Red points represents the necessary points that forms the path.

than a predefined threshold, search region gets smaller excluding the last point of the segment. Segmentation is based on the first and last points in the current search region. This procedure is repeated for all the search regions in the entire trajectory.

- $Visvalingam\text{–}Whyatt$ $(V-W)$: This approach defines effective areas for each location in the trajectory according to the consecutive neighbours. Given the points in the original dataset, effective areas for them are calculated. At each iteration, any point in the dataset with the smallest effective area is removed from the trajectory.
- $Douglas - Peucker$ $(D-P)$: It is a popular approach that produces a highly accurate line as an output. However, it has a high time complexity. D-P has a global routine in which the whole user trajectory is considered during the data reduction procedure. It takes the first and last points of the trajectory and draws a line. The farthest point form the line is selected. Then, a line is defined based on the first and farthest points. In this range, any point lower than a predefined threshold from the line is removed. This procedure is repeated for all the locations in the trajectory.

All the mathematical details about these techniques are provided in [15]–[19]. We deploy these techniques to prepare the dataset (i.e., reduced version) for the next step which is user mobility behaviour.

B. Learning user mobility behaviour

Objective: Given the modified trajectory of the user b $[C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow ... \rightarrow C_m]$, we plan to predict the future trajectory of the user for the next m time steps $[C_{m+1} \rightarrow C_{m+2} \rightarrow C_{m+3} \rightarrow ... \rightarrow C_{m+m}]$.

In this section, having the prepared movement history as prior knowledge, we feed this data to an RNN-based model to predict the future trajectory of user. The core concept is to take the user past locations and investigate the correlation between them in order to find repetitive patterns and predict the future trajectory.

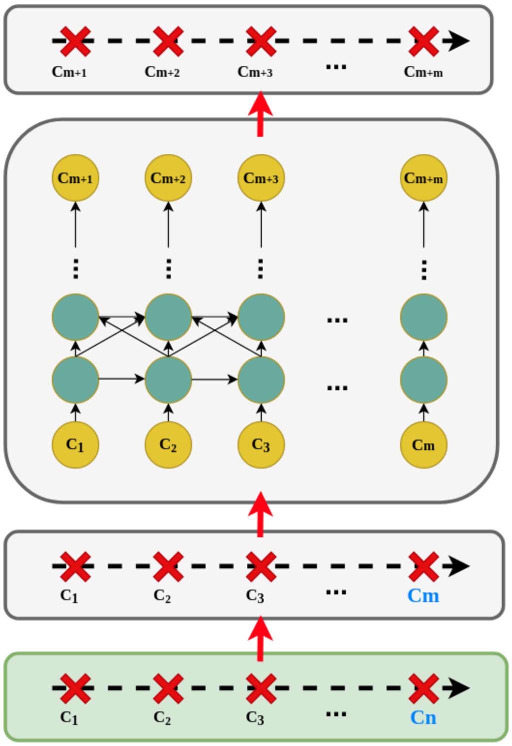


Fig. 2. Overall procedure of the proposed method.

For an RNN, given $C_b(t)$ as the past trajectory of the user b at time step t, hidden layer function $h(t)$ and output vector $y(t)$ are obtained by

$$h(t) = tanh(b + Wh(t-1) + UC_b(t)), \tag{1}$$

$$y(t) = softmax(b_1 + Vh(t)), \tag{2}$$

where W, U, V are weight matrices and b, b_1 are bias vectors [20].

The overall steps to process the prior information of user past location with an RNN are summarized as follows:

Step 1: We take the prepared data $[C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow ... \rightarrow C_m]$, which is the past mobility trajectory of the user, as the input of the mobility model.

Step 2: We initialize the weights and biases in the neural network.

Step 3: For the forward direction, at each step, we complete the forward pass of the network for hidden layers.

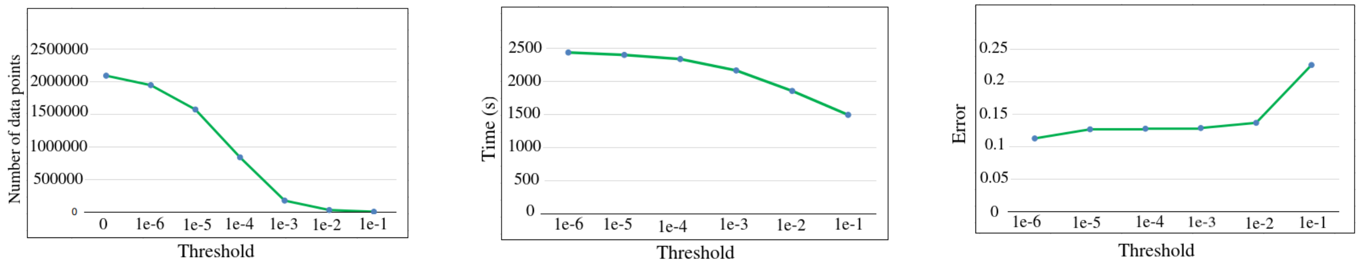
Step 4: We complete the forward pass for the output layer.

Step 5: We evaluate the prediction error.

Step 6: The calculated error is backpropagated to updated the parameters of the network.

Step 7: We have the future trajectory of the user for the next m time steps ahead $[C_{m+1} \rightarrow C_{m+2} \rightarrow C_{m+3} \rightarrow ... \rightarrow C_{m+m}]$.

Unfortunately, RNNs face the major problem of vanishing or exploding gradients in which the model error increases



(a) (b) (c)

Fig. 3. Impact of different thresholds on (a) data reduction, (b) execution time and (c) model performance.

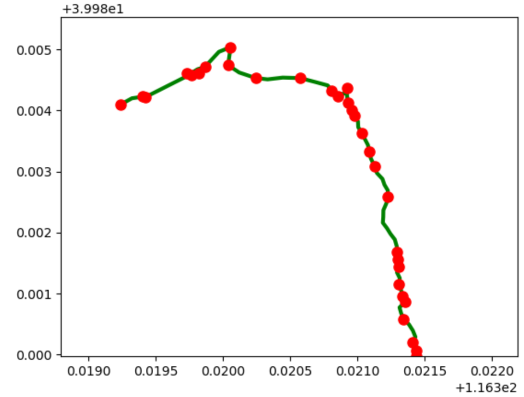


Fig. 4. Applying line simplification technique to the user trajectory to choose the necessary data points.

dramatically after a number of iterations. To solve this issue, gated RNNs are introduced including GRUs and LSTMs [20].

The key difference is the fact that GRU adds two new gates to the network: update $u(t)$ and reset $r(t)$ gates. These gates help the model to only keep the useful information about user past locations. The activation layer function in this case is given by

$$h(t) = u(t-1)h(t-1) + (1-u(t-1)) \qquad (3)$$
$$\sigma(b + UC_b(t-1) + Wr(t-1)h(t-1)),$$

where $u(t)$ and $r(t)$ are update and reset gates.

In LSTM, the main difference is adding a cell state unit $s(t)$ and a forget $f(t)$ gate to the model. In this case, the activation layer function is expressed as

$$h(t) = tanh(s(t))q(t), \qquad (4)$$

where $s(t)$ is cell state unit and $q(t)$ is LSTM output.

Fig. 2 represents the overall procedure of the proposed approach. First, we take the raw data with the length n and convert it into a optimized data with the length of m using line simplification techniques. Next, we feed the prepared data to the RNN-based pattern learning step. Finally, we have the predicted future trajectory of the user in the output.

IV. SIMULATION RESULTS

In this section, we conduct a series of experiments to analyze the impact of different line simplification methods on user mobility learning. We used a GPS dataset called "Geolife" for our experiments [21]. This dataset contains GPS trajectories (i.e., longitude and latitude) for 182 users. Geolife was collected over a period of almost five years (2007-2012). The length of trajectory (amount of available data) varies from user to user. We use these trajectories as movement history (i.e., prior knowledge) to learn the mobility behaviour of the user in order to make future location prediction. We employed Keras library (i.e., open source library in Python) to conduct our simulations, using an Intel core i7-6700k CPU, a 32 GB RAM and 4.00 GHz. In the following, first, we discuss the results regarding data reduction step. Then, we investigate the impact of data reduction on mobility learning.

TABLE I
APPLYING SIMPLIFICATION METHODS TO USER TRAJECTORY.

Methods	Threshold	Reduction	Time (s)
V-W	$1e-4$	2087853 → 121294	4.70
R-W	$1e-4$	2087853 →670009	10.53
nth Point	$n = 3$	2087853 →695952	0.03
Lang	$1e-4$	2087853 →21420	5.72
D-P	$1e-4$	2087853 → 837544	2239.16

A. Data reduction

Here, we chose the trajectory of the user number 153 for a period of approximately five years (from July 2007 to June 2012). Table I summarizes the results of applying simplification methods to the user trajectory. We considered $n = 3$ for the naive algorithm and the same threshold for the other methods. The original data contains $2,087,853$ data points. The threshold for each method, amount of the reduced data and execution time for each method are presented in Table I. As is shown, all the methods significantly reduced the amount of original data. D-P has the highest time complexity in comparison with the other algorithms. For D-P algorithm, worst case scenario time complexity is $o(n^2)$.

Fig. 3 depicts the impact of different thresholds of data reduction algorithms on three parameters. Fig. 3(a) shows how data reduction procedure works as we increase the threshold value. It is shown that data reduces noticeably with higher thresholds. In Fig. 3(b), we can see the effect of different thresholds on execution time (i.e., data reduction and mobility learning steps). Logically, as we increase the threshold and obtain smaller dataset, the execution time reduces significantly as well. Lastly, Fig. 3(c) demonstrates that how data reduction affects model performance (i.e., loss value of the learning algorithm). It is evident that reducing data for some thresholds does not result in model performance degradation. However, for higher thresholds (e.g., 1e-2, 1e-1), data reduction has disruptive effect on the performance. It is reasonable due to the fact that too much data is removed and much lower data is fed to the learning algorithm.

Fig. 4 shows a trajectory based on geographical coordinates (i.e., latitude and longitude). It represents the result of applying data reduction technique to choose the necessary data points. Here, we applied D-P algorithm.

B. Mobility learning

Here, we chose a part of the trajectory of the user number 153 for a period of approximately 13 months. We deployed a network composed of 3 layers with 100 neurons in each layer. Also, we used a learning rate of 0.001 and deployed Adam optimizer. Table II presents the impact of each data reduction algorithm on the performance of user mobility learning. We set the value of the thresholds in a way that the reduced data has almost the same size for all the line simplification methods. This is due to the fact that learning performance is highly correlated to the amount of data. When we have approximately the same size of data (length of trajectory) as input for the learning algorithm, we can compare the effect of different data reduction approaches on the overall performance. Time refers to the execution time of the both data reduction and learning algorithm. In Table II, mean square error (MSE) for the RNN model is reported. We chose this metric since our work is a regression-based task. Considering x, $y(x)$ and $\hat{y(x)}$ as orderly input data, real output and estimated output, for m samples out of k sample, mean square error is calculated by

$$MSE_m(\hat{y}) = 1/k \sum_{i=1}^{m} (y(x) - \hat{y(x)})^2. \tag{5}$$

This metric gives us an idea of how well the predictor works. Obviously, lower values of MSE is better and in an ideal situation MSE is equal to zero. As is shown, for this user, mobility learning without any data reduction results in 0.16 error loss value in 2245 seconds. Applying data reduction methods, D-P and Lang obtain respectively the best and worst performances with 0.12 and 0.23. It is evident that exploiting a proper data reduction method, not only does not cause performance degradation but also it may effectively improve it. This implies that during data reduction phase, some of the noisy data is removed as well. However, time complexity of D-P is noticeably higher than other methods. It is worth noting, the results may be different for a another user with a different mobility behaviour.

TABLE II
IMPACT OF DIFFERENT PREPROCESSING TECHNIQUES ON MOBILITY MODEL PERFORMANCE.

Methods	Threshold	Reduction	Time (s)	Error
No reduction	–	–	2245	0.16
V-W	$52e-6$	28163	466	0.19
R-W	$85e-6$	28185	432	0.22
nth Point	$n=5$	28111	448	0.17
Lang	$58e-6$	28150	471	0.23
D-P	$16e-5$	28115	582	0.12

Next, we want to investigate the impact of each line simplification approach as a prepossessing step for the three mobility learning models. Table III summarizes model loss values and execution times for RNN, GRU and LSTM. It is shown that LSTM model achieves the lowest error value among other mobility models. We can see that using data reduction techniques results in small performance degradation and in some cases even performance improvement but it can remarkably reduce execution time.

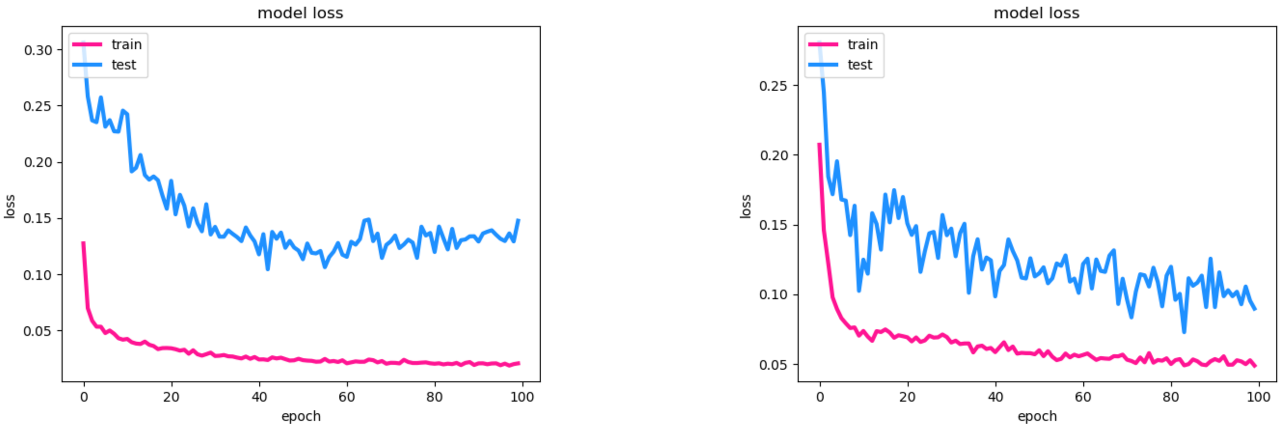
Fig. 5 shows train and test performances of the mobility learning model without (Fig. 5(a)) and with (Fig. 5(b)) data preprocessing. Here, D-P algorithm is deployed as reduction technique to remove the irrelevant data. As is shown, both train and test errors dramatically decrease after almost 40 epochs. It is noticeable that the average test error is much lower when using prepared data. Also, we can see that the model does not overfit and trains rapidly.

V. CONCLUSION

In this paper, we investigated several line simplification approaches as a potential preprocessing step for a mobility learning model in mobile networks. We proposed to deploy a line simplification method to reduce the irrelevant data from a big dataset and keep only the appropriate data for the mobility learning model. We analyzed the performance of the five data reduction techniques including nth point, R-W, Lang, V-W and D-P. Then, we used the reduced version of data for learning user mobility behaviour. We exploited RNN, LSTM and GRU models to find the repetitive patterns in user's movement history and predicted future trajectory of the user. Simulations results showed the effectiveness of the data reduction step. Using reduced data results in mobility model performance improvement. Moreover, it can remarkably reduce the execution time of the train and inference phases since we are dealing with smaller dataset. D-P algorithm produced the lowest mobility model error performance and the highest time complexity with respectively 0.10 and 733s while the model loss and execution time using the original dataset results in orderly 0.13 and 3057s. Among learning algorithms, LSTM model results in the most accurate trajectory prediction.

REFERENCES

[1] P. V. Klaine, M. A. Imran, O. Onireti, and R. D. Souza, "A Survey of Machine Learning Techniques Applied to Self-Organizing Cellular Net-



(a) Without data reduction.

(b) With D-P.

Fig. 5. Performance of LSTM-based model for train and test data.

TABLE III

IMPACT OF DIFFERENT SIMPLIFICATION TECHNIQUES ON THREE MOBILITY MODELS.

Method	RNN Error	Time (s)	GRU Error	Time (s)	LSTM Error	Time (s)
No reduction	0.16	2245	0.16	4616	0.13	3057
V-W	0.19	466	0.17	947	0.14	665
R-W	0.22	432	0.18	945	0.16	673
nth Point	0.17	448	0.13	932	0.11	666
Lang	0.23	471	0.20	961	0.19	680
D-P	0.12	582	0.11	1030	0.10	733

works," *IEEE Communications Surveys Tutorials*, vol. 19, Fourthquarter 2017.

[2] Cisco. [Online]. Available: https://www.cisco.com/annual-internet-report/white-paper-c11-741490.html

[3] Y. Zheng, "Trajectory Data Mining: An Overview," *ACM Trans. Intell. Syst. Technol.*, vol. 6, May 2015.

[4] R. Di Taranto, S. Muppirisetty, R. Raulefs, D. Slock, T. Svensson, and H. Wymeersch, "Location-Aware Communications for 5G Networks: How location information can improve scalability, latency, and robustness of 5G," *IEEE Signal Processing Magazine*, vol. 31, Nov 2014.

[5] N. Bui, M. Cesana, S. A. Hosseini, Q. Liao, I. Malanchini, and J. Widmer, "A Survey of Anticipatory Mobile Networking: Context-Based Classification, Prediction Methodologies, and Optimization Techniques," *IEEE Communications Surveys Tutorials*, vol. 19, thirdquarter 2017.

[6] G. Xu, S. Gao, M. Daneshmand, C. Wang, and Y. Liu, "A Survey for Mobility Big Data Analytics for Geolocation Prediction," *IEEE Wireless Communications*, vol. 24, February 2017.

[7] L. N. Balico, A. A. F. Loureiro, E. F. Nakamura, R. S. Barreto, R. W. Pazzi, and H. A. B. F. Oliveira, "Localization prediction in vehicular ad hoc networks," *IEEE Communications Surveys Tutorials*, vol. 20, Fourthquarter 2018.

[8] S. Qiao, D. Shen, W. Xiaoteng, N. Han, and W. Zhu, "A Self-Adaptive Parameter Selection Trajectory Prediction Approach via Hidden Markov Models," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 16, 02 2015.

[9] A. Nadembega, A. Hafid, and T. Taleb, "A Destination and Mobility Path Prediction Scheme for Mobile Networks," *IEEE Transactions on Vehicular Technology*, vol. 64, June 2015.

[10] V. Kulkarni, A. Mahalunkar, B. Garbinato, and J. D. Kelleher, "On the Inability of Markov Models to Capture Criticality in Human Mobility," *CoRR*, vol. abs/1807.11386, 2018. [Online]. Available: http://arxiv.org/abs/1807.11386

[11] A. B. Adege, H. Lin, and L. Wang, "Mobility predictions for iot devices using gated recurrent unit network," *IEEE Internet of Things Journal*, vol. 7, Jan 2020.

[12] C. Wang, L. Ma, R. Li, T. Durrani, and H. Zhang, "Exploring trajectory prediction through machine learning methods," *IEEE Access*, vol. PP, 07 2019.

[13] Y. Xing, C. Lv, and D. Cao, "Personalized vehicle trajectory prediction based on joint time-series modeling for connected vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, 2020.

[14] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the Next Location: A Recurrent Model with Spatial and Temporal Contexts," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16, 2016.

[15] K. Reumann and A. P. M. Witkam, "Optimizing curve segmentation in computer graphics," 1974.

[16] D. H. Douglas and T. K. Peucker, *Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature*. John Wiley & Sons, Ltd, 2011, ch. 2, pp. 15–28.

[17] T. Lang, "Rules for the robot draughtsmen." he Geographical Magazine, 42(1): 50–51, 1969.

[18] M. Visvalingam and J. D. Whyatt, "Line generalisation by repeated elimination of points," *The Cartographic Journal*, vol. 30, 1993.

[19] W. Shi and C. Cheung, "Performance evaluation of line simplification algorithms for vector generalization," *The Cartographic Journal*, vol. 43, 2006. [Online]. Available: https://doi.org/10.1179/000870406X93490

[20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016.

[21] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W.-Y. Ma, "Understanding transportation modes based on gps data for web applications," *ACM Transaction on the Web*, vol. 4, January 2010.