MoGAN: GAN based Next PoA Selection for Proactive Mobility Management

Boyun Jang
Dept. of Artificial Intelligence
Sungkyunkwan University
Suwon, Republic of Korea
boyunj0226@skku.edu

Syed M. Raza
Dept. of Electrical and Computer Engineering
Sungkyunkwan University
Suwon, Republic of Korea
s.moh.raza@skku.edu

Moonseong Kim
Dept. of Liberal Arts
Seoul Theological University
Bucheon, Republic of Korea
moonseong@stu.ac.kr

Hyunseung Choo
Dept. of Electrical and Computer Engineering
Sungkyunkwan University
Suwon, Republic of Korea
choo@skku.edu

***Abstract*—Current reactive mobility management in cellular networks becomes a bottleneck for ultra-low latency 5G services and severely degrades the QoS. To satisfy the ultra-low latency requirement of 5G services, proactive mobility management is essential where next PoA of the user is predicted with minimal error. Recent studies have used different deep learning algorithms for this purpose, but their results are unacceptable in real networks due to low accuracy. This paper exploits the distributional learning capability of Generative Adversarial Network (GAN) to propose MoGAN for the prediction of user's next PoA. The generator in MoGAN uses Gated Recurrent Unit to learn the distribution of time-series data and generates the next PoA. Meanwhile, the discriminator evaluates the generated output against the real data to determine its correctness. The model is trained in adversary mode by using the output from the discriminator. The dataset utilized in training and evaluation is collected from one of the university campuses, and the results show 96.33% of prediction accuracy, which is 5% higher than the previous study. Furthermore, MoGAN is more robust under limited data conditions, as it achieves above 90% accuracy with only 50% of the dataset.**

***Index Terms*—Generative Adversarial Network, Gated Recurrent Unit, Deep learning, Mobility management, Prediction**

I. Introduction

5G mobile networks are envisioned to enable Ultra Reliable Low Latency Communication (URLLC) for high performance services like Virtual Reality, 8K video streaming, and autonomous vehicles. It is achieved through ultra-dense deployment of network nodes to increase coverage, bandwidth, and frequency reuse. Consequently, handovers between cells are substantially increased, and at each handover there are multiple surrounding Point-of-Attachments (PoAs) [1]. This makes the correct selection of next PoA challenging, as in current user assisted reactive mobility management system, the user selects the next PoA based on the highest received signal strength from the surrounding PoAs. Such selection criteria increase the number of handovers which in turn severely effects the network performance and user QoS, and consume additional bandwidth for control signaling.

Proactive mobility management is an effective way to manage handovers and attain URLLC in a cellular network where PoA changes frequently [2]. High accuracy prediction of next PoA of a mobile device is a mandatory requirement of proactive mobility management, and it also enables network system to manage resources more effectively. Former studies have predicted next PoA by adapting various conventional methods like Markov model [3]. A more recent study has exploited the sequential and time-series characteristics of the mobile device movement, and used Deep Learning (DL) models such as Long Short-Term Memory (LSTM) to predict the next PoA with 91% accuracy [4]. LSTM is based on Recurrent Neural Network (RNN), and it captures the relation between data points in long sequences by solving the vanishing gradient problem of RNN. However, LSTM performance is limited to the availability of large amount of data for learning, which is not always the case in cellular networks.

Generative Adversarial Network (GAN) is a framework consists of two adversarial neural network structures, namely Generator and Discriminator [5]. GAN exhibits several features which are induced by its competitive structures. Each neural structure in GAN gains strength by receiving increasingly useful feedback from its growing adversaries. Original purpose of GAN based models is to generate data, therefore the feedback in GAN is based on the distribution of the generated data, that is, GAN is trained to reflect the whole distribution of the data. Moreover, due to various forms of Deep Neural Network (DNN) like RNN and Convolutional Neural Network (CNN), various kinds of data can be used in GAN by changing the DNN structure in the generator and discriminator. GAN based models have shown exceptional

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.2019-0-00421, AI Graduate School Support Program(Sungkyunkwan University)) and grant funded under the GITRC support program(IITP-2020-2015-0-00742), and also supported by the National Research Foundation of Korea (NRF-2020R1A2C2008447).

978-1-7281-6992-7/20/$31.00 ©2020 IEEE

performance in generating data like images and sounds, but it can also be applied for other usages like Intrusion Detection System (IDS) [6] and recommendation system [7].

This paper overcomes LSTM limitation by proposing GAN based mobility prediction (MoGAN) model that utilizes the adversarial and data distributional learning characteristics of GAN to predict next PoA of a mobile device. Generator in MoGAN is based on Gated Recurrent Unit (GRU) and predicts the next item in the data sequence (i.e., PoA), whereas discriminator consists of fully connected layers and classifies between real and generated sequential data. In addition to general adversarial training method in GAN, we propose an additional feedback to the generator for steady convergence of the generated data towards real data. MoGAN is trained on a real wireless network dataset that is collected for little over two months at the university campus, and in this paper we term it as Campus Mobility Dataset (CMD). Testing results show that for sequence length of 31, MoGAN achieves highest accuracy of 96.33% with 7% and 5% increment over vanilla GAN and LSTM model [4], respectively. Moreover, MoGAN predicts next PoA with better accuracy than LSTM even when only half of the data is used for training and testing.

II. Deep Learning Approaches for Prediction

A. LSTM model

RNN is the neural network structure for supervised learning that captures the features of continuous data for prediction. Between the input and output layers, additional hidden layers learns properties of sequential data recurrently to predict the next step of sequence. However, it has a long-term dependency problem where the past information exponentially decays as the sequence length becomes long. LSTM solves this problem by introducing cell state which saves more information of past sequences [8]. Three gates (forget, input and output) control the cell state and protect the past information. LSTM based prediction models have shown good performance for long sequential data and that is why it has been widely used in different applications such as recommendation system [9].

Complex structure of LSTM cell makes it computationally expensive. GRU simplifies the LSTM cell by combining cell state and hidden state, and uses two gates instead of three [10]. Similar to LSTM, GRU also has additional gate to control the memory of past sequences. However, GRU has fewer parameters to calculate comparing to LSTM and this makes GRU simpler and computationally less expensive. Moreover, GRU shows even better performance than LSTM for less complex datasets. This enables GRU to be used for applications with less complex data like sentiment analysis [11].

B. GAN as a Prediction Model

Initially suggested vanilla GAN has two competing structures, generator and discriminator, which are trained by minimax game method as shown in Fig. 1. Noise vector without any information about real data is used as generator input. This noise vector goes through some layers in generator to make generated data. Discriminator tries to classify between real and generated data. If real data comes as an input, discriminator tries to make the output close to 1. Otherwise the output is desired to be close to 0. Because the generator wants to create fake data similar to the real one, it is trained to make the output of discriminator close to 1 with generated data. When the output of discriminator for every data becomes 0.5, it means generator has succeeded to confuse discriminator by making real-like data.

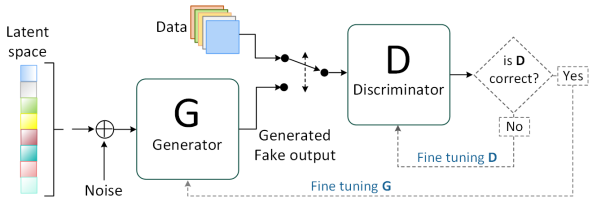

Fig. 1. GAN architecture.

To make the real-like data, generator is trained to learn the distribution of real data. Therefore, GAN is often used as a generating model, especially in case of applications related to image enhancement and augmentation [12]. A main characteristic of GAN is that it has two competing structures giving useful feedback to each other, and each structure can be used in various ways to achieve different objectives. For example, discriminator can have good classification ability, that is, discriminator can be used as a good classification model. Along this line, GAN is used as a way to detect malicious traffic data for Intrusion Detection System (IDS) in [6]. Similar to discriminator, the generator can also have different neural network models such as RNN to make predictions based on sequential data for applications such as recommendation system [7] and stock market [13].

III. MoGAN

A. Architecture

The proposed MoGAN model predicts next PoA of the mobile device by learning from the data consisting of sequences of previous PoA connections. As the proposed model is based on GAN, it consists of two neural network structures, generator and discriminator. In a complete sequence of PoAs $X = \{x_1, x_2, ..., x_{n-1}, x_n\}$, the model predicts x_n by learning from the sequence of previous PoAs $X_p = \{x_1, x_2, ..., x_{n-1}\}$. Generator predicts next PoA x'_n which it thinks most likely to come after X_p. Discriminator then classifies between X and partially generated sequence $X' = \{x_1, x_2, ..., x_{n-1}, x'_n\}$, which is the combination of X_p and x'_n. The overall structure of the proposed MoGAN is shown in Fig. 2.

The generator generates x'_n by learning the properties of X_p. RNN based structures are the best suited neural network structures to learn the properties of sequential or time-series data. Between LSTM and GRU (i.e., two RNN based structures), MoGAN uses GRU in the generator because of its equivalent performance to LSTM in case of small datasets and lower computational overhead. Moreover, use of computationally expensive LSTM in already computation intensive

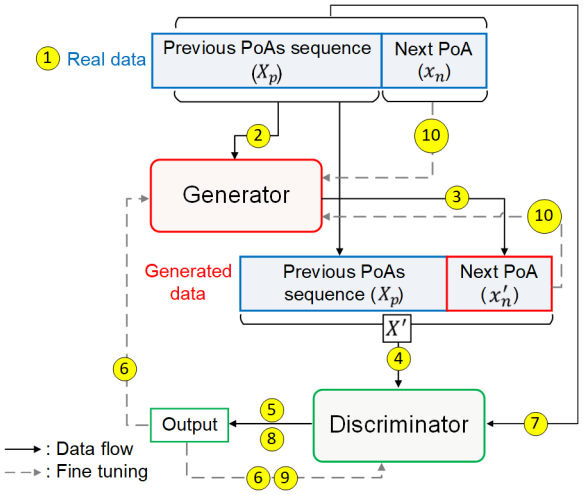

Fig. 2. MoGAN architecture and data flow.

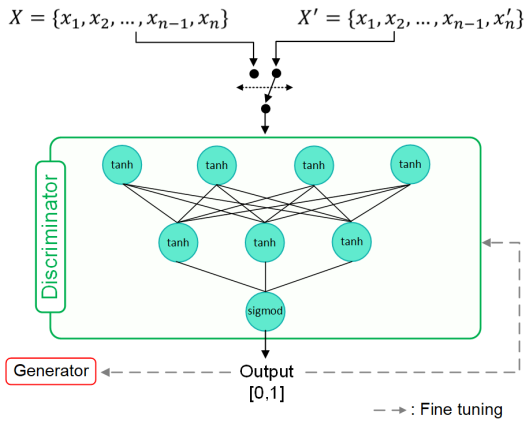

Fig. 4. Discriminator structure in MoGAN.

GAN causes unstable learning. GRU in the generator consists of single layer and has many-to-one structure, which takes X_p as input, as shown in Fig. 3. Output from GRU is fed to a fully connected layer with softmax function to get x'_n. During the training of MoGAN, x'_n is concatenated with X_p to get X'.

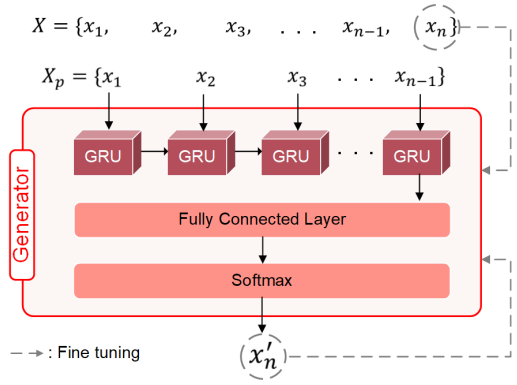

Fig. 3. Generator structure in MoGAN.

Discriminator classifies between real sequence X in the data and partially generated sequence X'. To classify between real and generated sequences, neural network structure in the discriminator must recognize all the properties of the sequences. To this end, fully connected neural network performs better comparing to RNN based structures [14]. Thus, MoGAN discriminator consists of fully connected layers as shown in Fig. 4. The input layer takes sequences X or X' as input, and after passing through single hidden layer the output layer gives a value between 0 and 1. To summarize, MoGAN architecture consists of a GRU based generator and a fully connected neural network based discriminator.

B. Data Preprocessing

Each PoA in the sequences holds the same preference, that is, all the PoAs must be treated equally. Transferring each PoA into One-hot vector is useful in this aspect, where they have their own N-dimensional vector when there are total N points in the data. The index in the vector equivalent to the ID of the PoA has the value one, and all other values are zero. This allows the index of the highest value in the vector to identify the PoA. For example, if there are five PoAs in the data, the vector of PoA three can be represented as $\{0, 0, 1, 0, 0\}$, while vector of PoA five is $\{0, 0, 0, 0, 1\}$. After training, if the generator generates next point as $\{0.0, 0.2, 0.1, 0.6, 0.1\}$, then it can be identified as PoA four, as the index of highest number in the vector is four.

The movement history of a mobile device can be represented as a sequence of PoAs, and the length of sequences are different from each other representing the time each mobile device has spent in the network. As the next PoA is predicted based on learning the previous sequence of PoAs, a certain length of sequence is required to properly learn and correctly predict the next PoA. In particular, sequences must have equal length for the training because of two reasons: to filter out short sequences that are not fit to predict the next PoA appropriately, and to determine the necessary sequence length for high accuracy prediction. Suppose, sequences with length five are selected to train the MoGAN, then actual length of sequences is six, where former five PoAs represent X_p and last 1 PoA is x_n. Sequences with lengths less than six are filtered out, and sequences longer than six are divided into several sequences through one step right shift. For example, three sequences can be derived from a sequence of length eight: first sequence from 1st PoA to 6th PoA, second one from 2nd PoA to 7th PoA, and third from 3rd PoA to 8th PoA.

C. Training

The training process of MoGAN is divided into two steps: Minimax step (Step 1) and generator training step (Step 2). Throughout the training, binary cross entropy is used as the error function (1), where y is expected value, and y' is predicted value. In step 1, MoGAN is trained through minimax game which is similar to the training in vanilla GAN. In particular, role of discriminator is to maximize the result of X and minimize it for X' (2). X' is a combination of X_p and

$x'_n = G_\theta(X_p)$ and it is represented as $X_p + G_\theta(X_p)$ in (2), (3), and (4). Contrary to discriminator, the role of generator is to maximize the discriminator result for X' (3). A combined function representing this Minimax game is shown in (4). G_θ and D_ϕ are the weight values of generator and discriminator, respectively.

$$H(y, y') = -\frac{1}{N}\sum_{i=1}^{N}(y_i log(y'_i) + (1 - y_i) log(1 - y'_i)) \quad (1)$$

$$D\colon \max_{D_\phi}[H(0, D_\phi(X)) + H(1, D_\phi(X_p + G_\theta(X_p)))] \quad (2)$$

$$G\colon \min_{G_\theta}[H(1, D_\phi(X_p + G_\theta(X_p)))] \quad (3)$$

$$\min_{G_\theta}\max_{D_\phi} = H(0, D_\phi(X)) + H(1, D_\phi(X_p + G_\theta(X_p))) \quad (4)$$

Unlike the vanilla GAN, the purpose of generator in MoGAN is to predict the next PoA from the given sequence and not to generate fake data resembling the real one. This means that from X_p generator output $G(X_p)$ should be same as x_n. In minimax training, the generator is oblivious of x_n and only works towards deceiving the discriminator. Additionally, the inability of generator to generate appropriate next PoA x'_n affects the discriminator classification ability, and drifts the whole training process towards inconsistency. To address this issue, MoGAN generator is additionally trained (Step 2) with x_n information after the completion of Step 1. This allows generator to train to minimize the difference between x_n and x'_n, and stabilizes the overall training processes.

$$\min_{G_\theta}[H(x_n, G_\theta(X_p))] \quad (5)$$

Summary of overall training and data processing in MoGAN is shown in Fig. 2, where real sequence of PoAs X is split into X_p and x_n ①. X_p is fed into the generator as input ② to generate the predicted next PoA x'_n ③. The combination of X_p and x'_n, X', is fed into the discriminator ④ for classification. Based on the output ⑤, the discriminator is trained to minimize this output towards 0 while the generator is trained to maximize it towards 1 ⑥. Next, X is fed into the discriminator ⑦ which tries to maximize the output towards 1 ⑧⑨. This completes the Minimax game based training in Step 1, and in Step 2 the generator is trained with the difference between x_n and x'_n ⑩. The completion of Step 2 makes one epoch of the MoGAN training. Algorithm 1 further details the complete training of MoGAN.

Algorithm 1: The training procedure of MoGAN

Initialize: Number of total epoch n, number of Step 2 per epoch α, randomly initialized weights θ, ϕ *for* G_θ, D_ϕ

input : $X = \{x_1, x_2, ..., x_{n-1}, x_n\}$

Error function ← binary cross entropy H

for n **do**

- $G_\theta(X_p)$ predicts next PoA x'_n
- X' ← Combine X_p with x'_n
- D_loss_real ← Get loss value from D for real data $H(0, D_\phi(X))$
- D_loss_fake ← Get loss value from D for generated data $H(1, D_\phi(X'))$
- Update ϕ to maximize D_loss_real + D_loss_fake
- G_loss_Step1 ← Get loss value from G for $H(1, D_\phi(X_p + G_\theta(X_p)))$
- Update θ to minimize G_loss_Step1
- **for** α **do**
  - G_loss_Step2 ← Get loss value from G for $H(x_n, G_\theta(X_p))$
  - Update θ to minimize G_loss_Step2

IV. Evaluations

A. Setup

We have trained and tested MoGAN using CMD, which is collected from the wireless network of Intelligent ICT Convergence Research Center in Pangyo, Republic of Korea [4]. The wireless network consists of 12 APs and it is used by 289 users in the campus and each user is connected to only one AP at a given time. The network controller saves a handover log based on the user mobility in the campus. The acquired log files consist of mobile device MAC addresses and the IDs of the handover source and destination APs. Sequences of mobile devices movement are created using log entries where destination AP ID in one entry is same as the source AP ID in other entry for the same MAC address.

For each experiment, MoGAN generator consists of one GRU layer with 512 nodes and one dense layer with 12 nodes. The dense layer uses softmax as an activation function to generate next PoA in One-hot vector format resembling the real next PoA. The neural network in the discriminator consists of three layers. The input layer and hidden layers consists of 128 and 64 nodes, respectively, and uses tanh activation function due to its superior performance in binary classification between two classes [15]. The output layer consists of a single node with sigmoid activation function whose output is between 0 and 1.

MoGAN training uses Adam optimizer with 0.001 learning rate, and binary cross entropy as the loss function. All the pre-processed sequences in CMD are shuffled and randomly separated into training and test data with the ratio 7:3. In experiments, the number of epochs, sequence length, and the training iterations of Step 1 and Step 2 per epoch are set as 4,000, 31, and 1, respectively, unless stated otherwise.

B. Experiments

We have evaluated MoGAN performance in terms of next PoA prediction accuracy for different sequence lengths, number of epochs, Step 1 and 2 training iterations, and amount

of data used for training. Also, MoGAN is compared against vanilla GAN and stacked LSTM [4] to outline the gain in accuracy achieved by MoGAN in reference to time and data consumed in training. Each result presented in this paper is an average of ten experiments which are conducted on a system with Intel Core i7-9700K CPU, two NVIDIA GeForce RTX 2060 SUPER graphic cards, and 32GB RAM.

Sequence length defines the movement pattern of the mobile device, and intuitively it is easier to determine the pattern with longer sequence length. Knowing the mobile device movement history allows the generator to determine the movement pattern and predict the next PoA with higher accuracy. The results in Fig. 5 show the next PoA prediction accuracy of MoGAN and vanilla GAN (i.e., MoGAN without Step 2 in training) for different sequence lengths, where highest accuracy achieved by MoGAN and vanilla GAN is 96.33% and 88.17% for sequence lengths of 31 PoAs and 27 PoAs, respectively. This quantifies and highlights the prediction accuracy gain achieved by proposed Step 2 in the training of MoGAN. It is note worthy that sequence length for highest prediction accuracy is relative to the characteristics of dataset, but it is absolute that performance of MoGAN is better than of vanilla GAN due to its superior training methodology.

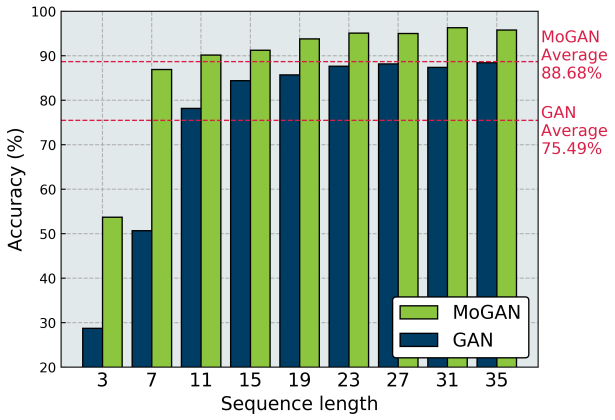

Fig. 5. Next PoA prediction accuracy comparison between MoGAN and vanilla GAN for different sequence lengths.

As we have demonstrated the significance of Step 2 in MoGAN training, it is valuable to analyze the effects of increased number of Step 2 training (α) in each epoch (i.e., after every Step 1 of training). The results in Fig. 6 illustrate the prediction accuracy achieved by different values of (α) in MoGAN for different sequence lengths. The accuracy achieved by different α values is roughly consistent with each other and no significance gain is attained. This indicates that additional training of generator with one iteration of Step 2 (i.e., α=1) is sufficient, and repeating Step 2 two or three times (i.e., α=2 and α=3) has insignificant effect.

We have compared MoGAN against state-of-the-art stacked LSTM model [4] over same dataset. For a sequence length of 31, MoGAN reaches the prediction accuracy of 96.33% in 4,000 epochs while consuming 190seconds in training

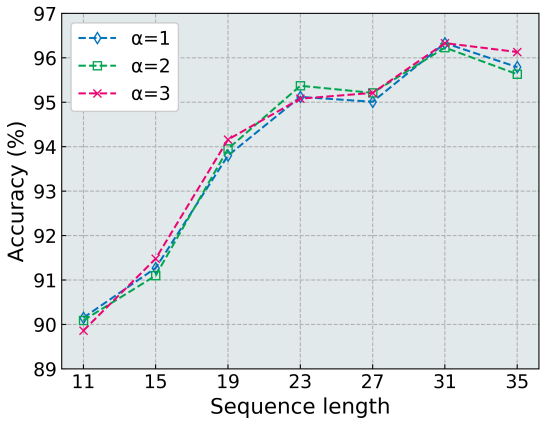

Fig. 6. Analysis of MoGAN with different iterations of Step 2 training (α) with increasing sequence length.

and testing. The results remain consistent when number of epochs increases, as shown in Fig. 7. Comparatively, LSTM attains prediction accuracy of 95.95% in 9,000 epochs. While analyzing only prediction accuracy between MoGAN and LSTM the difference is subtle, however, after factoring in the time taken by both models to achieve their respective maximum accuracy, it is evident that stacked LSTM takes more than double training time than MoGAN. It is a general assumption that supervised learning is done offline prior to system initiation, and the training time is not relevant. However, after certain time interval it is desirable to retrain the learning model, with newly accumulated date to incorporate new data features. Training time is highly important in this scenario as it defines the downtime of the prediction system, and MoGAN reduces this downtime by half in comparison to LSTM. Once the system is online, MoGAN and LSTM takes 5.85ms and 9.15ms, respectively, to predict next PoAs for 708 devices that aptly represent mobility prediction requirement of real cellular network at a given time. This 40% less prediction time by MoGAN shows its deployment feasibility in cellular networks and significance in maintaining service quality for ultra low latency applications.

One of the major limitations of supervised deep learning models is the dependency on data, which is not always readily available. Thus, it is desirable to compare and evaluate the performance of both the prediction models with limited amount of data. For the sequence length of 31, there are total 2,362 sequences in CMD, and while maintaining the training and test data ratio, we conducted the experiment by varying the number of sequences used from 20% to 100%. This means if 20% of sequences in the dataset are used, then only 472 sequences are used, where 330 sequences are for training and 142 sequences are for testing. Results in Fig. 8 depicts that with only 20% of data MoGAN achieves 86.52% prediction accuracy and it goes above 90% when 50% of data is used. In comparison, the performance of LSTM is much more affected with decrease in the training data. This result not only establishes MoGAN

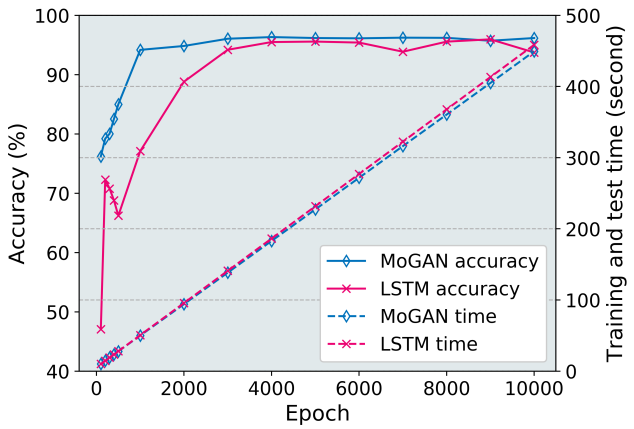

Fig. 7. Performance comparison between MoGAN and stacked LSTM in terms of next PoA prediction accuracy and time cost.

precedence over LSTM but also highlights the influence of adversarial learning and novel Step 2 in the training to achieve reasonably good results with limited amount of data and less computation. Additionally, it can be deduced that MoGAN will show even better performance with large and diverse datasets from cellular networks.

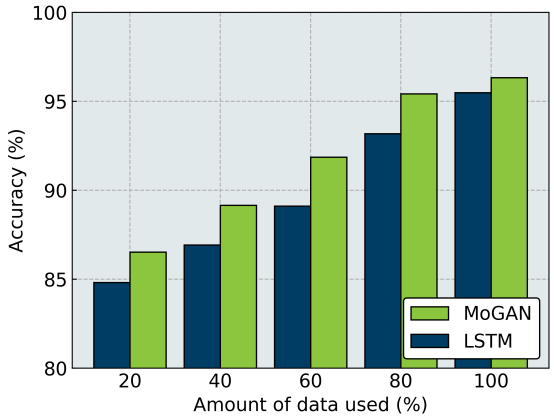

Fig. 8. MoGAN and LSTM performance comparison with limited data.

V. Conclusion

This paper proposes MoGAN, a GAN based next PoA prediction model for proactive mobility management in cellular and wireless networks. The generator in MoGAN uses GRU for predicting next PoA of mobile device, while discriminator tries to determine the difference between predicted next PoA and the real PoA through classification model. We also propose an additional training step (Step 2) in MoGAN, and results exhibit that it allows MoGAN to gain approximately 7% accuracy improvement over vanilla GAN. They also reveal that increasing number of Step 2 per epoch do not improve performance and a single Step 2 per epoch is sufficient. Comparison against state-of-the-art LSTM prediction model shows that prediction accuracy of MoGAN is slightly higher and it is attained in less than half the time. Moreover, with limited amount of data for training and testing, MoGAN outperforms LSTM and delivers relatively high accuracy with significantly reduced data. Overall, the proposed MoGAN model defines an improved method for data based prediction which can be used in other domains as well. We are currently working towards the improvement of MoGAN through other attention mechanisms than one-hot vector and test it with more diverse mobility datasets from cellular networks. Additionally, we look to extend MoGAN from single step to multiple step prediction for strategic placement of network resources.

Acknowledgment

Authors three and four are the co-corresponding authors.

References

[1] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5g: Ran, core network and caching solutions," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 3098–3130, 2018.

[2] M. Ozturk, M. Gogate, O. Onireti, A. Adeel, A. Hussain, and M. A. Imran, "A novel deep learning driven, low-cost mobility prediction approach for 5g cellular networks: The case of the control/data separation architecture (cdsa)," *Neurocomputing*, vol. 358, pp. 479 – 489, 2019.

[3] Q. Lv, Y. Qiao, N. Ansari, J. Liu, and J. Yang, "Big data driven hidden markov model based individual mobility prediction at points of interest," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 6, pp. 5204–5216, 2017.

[4] H. Yang, S. M. Raza, M. Kim, D. Le, V. Van Vo, and H. Choo, "Next point-of-attachment selection based on long short term memory model in wireless networks," in *14th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, 2020, pp. 1–4.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014, pp. 2672–2680.

[6] Z. Lin, Y. Shi, and Z. Xue, "Idsgan: Generative adversarial networks for attack generation against intrusion detection," *arXiv preprint arXiv:1809.02077*, 2018.

[7] H. Bharadhwaj, H. Park, and B. Y. Lim, "Recgan: Recurrent generative adversarial networks for recommendation systems," in *12th ACM Conference on Recommender Systems*, 2018.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, "Recurrent recommender networks," in *10th ACM International Conference on Web Search and Data Mining*, 2017, p. 495–503.

[10] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[11] L. Zhang, Y. Zhou, X. Duan, and R. Chen, "A hierarchical multi-input and output bi-GRU model for sentiment analysis on customer reviews," *Conference Series: Materials Science and Engineering*, vol. 322, 2018.

[12] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[13] K. Zhang, G. Zhong, J. Dong, S. Wang, and Y. Wang, "Stock market prediction based on generative adversarial network," *Procedia Computer Science*, vol. 147, pp. 400 – 406, 2019.

[14] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[15] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018.