Deep Mobile Path Prediction With Shift-and-Join and Carry-Ahead

Huigyu Yang, Syed M. Raza, Moonseong Kim, and Hyunseung Choo, *Member, IEEE*

***Abstract*—Importance of user mobility has rapidly increased in 5G due to reduced cell sizes, management of Multi-access Edge Computing (MEC), and ultra-low latency services. Reactive nature of existing management systems is a bottleneck, and it can be solved by building proactive systems that exploit temporal characteristics of time-series mobility data to predict long-term user movement (*i.e.*, path). However, user mobile path prediction with useable accuracy is a challenging task, particularly for lengthy target trajectories. This paper adopts general approaches to propose two models for predicting mobile path with high accuracy. Step Forward Iteration (SFI) model is based on recursive approach, whereas Encoder-Decoder (ED) model follows multi-output approach, and both the models use Long-Short Term Memory (LSTM) as the learning unit. Training and testing of these models is done on mobility datasets from the wireless network of Pangyo ICT Research Center, Korea and one of the Korean mobile operators. The experiment results show viability of the proposed models for leveraging mobile network management, as they outperform state-of-the-art GRU with attention (GRU-ATTN) and Transformer Network (TN) models. The highest prediction accuracies achieved for 3, 5, and 7 steps of target sequences (*i.e.*, predicted mobile path) in the campus dataset are 96%, 90%, and 87%, respectively.**

***Index Terms*—Mobile user path, multi-step prediction, recursive approach, multi-output approach, proactive mobility.**

I. Introduction

5G MOBILE networks provide new services with ultra-low latency through dense deployment of small cells and moving network/service functions at the network edge by using Multi-access Edge Computing (MEC). Current mobility management system uses the signal strength of nearby base stations to make handover decisions, however, multiple base stations in dense cell deployment can have high similarity in signal strengths. This makes handover decisions a challenging and critical task because erroneous handovers increase the load on system and degrades the Quality of Service (QoS). Moreover, service functions in MEC needs to migrate with the user mobility to maintain the service quality by reducing the transmission delay between user and service function. Service function migration is a challenging task as it is a time intensive operation and requires resource allocation in next MEC [1]. Prediction based proactive mobility management is one of the solutions to tackle these challenges, where next Point of Attachments (PoAs) of user mobiles are predicted in time-series. This reduces erroneous handovers and enables mobile networks to preemptively allocated resources in MEC for delay intolerant services like AI-assisted autonomous driving [2], [3]. Thus, proactive mobility management is essential for 5G mobile networks, however, predicting the next multiple PoAs with high accuracy is crucial for mobile networks performance.

Proactive mobility prediction is achieved through various conventional models. Markov-process based models show good prediction results, and one of studies has used it the model to capture spatiotemporal nature of user mobility [4], [5]. Another study exploits Hidden Markov Model to reduce the computation cost in addition to good prediction results [6]. These conventional mobility prediction approaches are not applicable in real mobile networks due to their limited prediction accuracy and high time complexity. Deep Learning (DL) models learn from given time-series data and have shown high prediction accuracy in real time. In particular, Long Short-Term Memory (LSTM) based DL model achieves better performance than machine learning models [7] and it has shown 91% accuracy in next PoA prediction from time-series data for proactive mobility management [8]. Single step (*i.e.*, next PoA) prediction is not sufficient for mobility management in dense cell deployment, as it does not provide sense of the path for effective handover decisions and optimal server selection in MEC for resource allocation. To overcome the limitations of single-step mobility prediction, multi-step user mobile path prediction is necessary.

User mobile path is a combination of multiple steps in time-series, where in each step a connection with PoA is established. Hence, user path prediction can be addressed as multi-step prediction problem. Multi-step prediction is generally achieved through either direct, recursive, or multi-output approach. These approaches can be realized through various DL models that have shown better performance than

Manuscript received 25 March 2022; revised 30 August 2022 and 3 December 2022; accepted 25 January 2023. Date of publication 6 February 2023; date of current version 9 June 2023. This work was partly supported by the Ministry of Education, Institute of Information and Communications Technology Planning and Evaluation (IITP), and the National Research Foundation (NRF), Korea, under the GITRC support program (IITP-2022-2015-0-00742), the ICT Creative Consilience program (IITP-2022-2020-0-01821), the High-Potential Individuals Global Training Program (IITP-2021-0-02132), and the mid-career support program (NRF-2020R1A2C2008447). The associate editor coordinating the review of this article and approving it for publication was J. Liu. *(Corresponding authors: Hyunseung Choo; Moonseong Kim.)*

Huigyu Yang is with the Department of Superintelligence Engineering, Sungkyunkwan University, Suwon 16419, South Korea (e-mail: huigyu@skku.edu).

Syed M. Raza and Hyunseung Choo are with the Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon 16419, South Korea (e-mail: s.moh.raza@skku.edu; choo@skku.edu).

Moonseong Kim is with the Department of IT Convergence Software, Seoul Theological University, Bucheon 14754, South Korea (e-mail: moonseong@stu.ac.kr).

Digital Object Identifier 10.1109/TCCN.2023.3242376

conventional solutions, and they take time-series data as input and provide multi-step prediction as output. DL models for multi-step prediction have their own technical challenges among which error accumulation is the most prominent and causes severe performance degradation. In error accumulation, the prediction error of first step causes larger error in the next step and this way errors accumulate till the last step. Prediction accuracy of the first step and length of predicted sequence are the major influencers of error accumulation, where high prediction accuracy of first step can reduce it substantially. Other important challenges in determining the performance of multi-step prediction DL models include the composition of time-series data for training and its collecting environment. The mobile path of users is obtained through their movement which is depending on environmental features such as regional characteristics and user behaviors [9]. Most of the studies train using synthetically created data through simulations which makes their results unreliable due to limited representation of real-world user mobility patterns in the synthetic data [10], [11]. Multi-step prediction using real world mobility data is presented in [12], and the results highlight the challenges and negative impact of incorrect data manipulation and preprocessing methods. For example, the generalization capability of a model diminishes due to data segregation per user and separated training by using individual dataset [7].

Aiming at the above problem, we propose Step-Forward Iteration (SFI) model using recursive approach and Encoder-Decoder (ED) model based on multi-output approach for user mobile path prediction. SFI model predicts single step result iteratively that is manipulated by data feeding layer to attain multi-step prediction. ED model performs two distinct networks jointly, where Encoder network converts input sequences to fixed-length vectors and Decoder network output predicted mobile path based on the vectors. LSTM is used as underlying learning unit in the models due to its talent in addressing long sequences of time-series data. These models are evaluated using identifier-based dataset collected from wireless network of a research center for two months and preprocessed. Through the experimental results, we discuss heterogeneous factors that influence performance and stability of SFI and ED models and present their architectural effectiveness in the mobile network management perspective. The major contributions of this paper are summarized as follows.

- A novel data processing algorithm tracks raw single-hop handover logs of each user in two live wireless networks mobility datasets, and stich them together to craft mobile path sequences of variable lengths. The length of such a path sequence is configurable during the extraction process to tolerate network scaling. To the best of our knowledge, our raw handover log and the dataset are the only publicly learnable resources [13] in this field.
- Selection of LSTM as an underlying spatial-temporal learning unit is to fit the characteristics and structure of created time-series sequences, and has best performance in comparison to other ML and conventional techniques.
- Proposed SFI model based on LSTM implements a novel data feeding layer that repeatedly exploits proposed shift-and-join method with predicted output of previous step to create input sequences for iterative multi-step prediction. It utilizes discounted significance of accumulation assigned to inputs during the training to reduce the accumulated errors in testing.
- Proposed ED model extends the iterative SFI model to multi-step carry-ahead prediction for further mitigation of accumulated errors in testing, and also utilizes discounted significance of accumulation assigned inputs in training.
- Comprehensive performance evaluation and comparison of the proposed SFI and ED models against state-of-the-art GRU with attention (GRU-ATTN) and Transformer Network (TN) models using preprocessed datasets. The results show that the proposed models have 7.09% maximum accuracy gain over target models for campus dataset, and for operator dataset it increases to 21.17%.

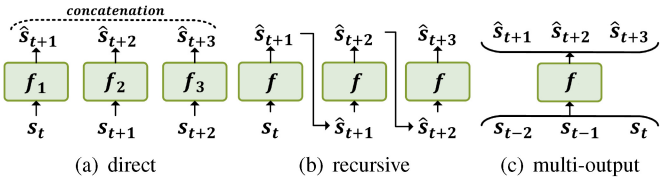


Fig. 1. Three different approaches for multi-step prediction.

The remaining paper is structured as follows. Section II explains three approaches for multi-step prediction with literature reviews. Section III first describes the collection and preprocessing of the dataset and then thoroughly discusses the two proposed models. Section IV shows experimental results and analyzes the performance of the proposed models. Conclusion and future works are presented in Section V.

II. Related Work

Multi-step prediction requires to predict next q elements of the sequence $\langle \hat{s}_{p+1}, \hat{s}_{p+2}, \ldots, \hat{s}_{p+q} \rangle$ from p elements of the sequence $\langle s_1, s_2, \ldots, s_p \rangle$ in time-series. Literature studies for the mobile path prediction generally follow one of the three general approaches for multi-step prediction namely, direct, recursive, and multi-output [14], [15]. As shown in Fig. 1, these three approaches are categorized based on their structural features and outputs of multi-step prediction results. For example, the direct approach-based prediction model in Fig. 1 predicts next three steps of a sequence by concatenating the outputs from multiple leaning models f_1, f_2, and f_3. In contrast, recursive approach-based prediction model uses a single learning model f_1, which iteratively utilizes the predicted output to generate next output of a sequence. Multi-out approach similarly uses a single learning model f_1 that maps multiple values of input sequence to output sequence simultaneously. Subsequent subsections give a detailed account of three multi-step prediction approaches with their related literature review.

A. Direct Approach for Multi-Step Prediction

Direct approach for multi-step prediction uses individual learning models for predicting each element in the target sequence. These models are trained to predict the next

single-step, and their results are concatenated in chronological order to output the multi-step prediction result. As only ground truth values are used in the input sequences of the models, prediction error of one model does not affect the next step and this avoids the error accumulation problem. However, temporal dependencies are major characteristics of time-series data that the direct approach-based models are unable to exploit due to the use of individual models. The computational cost of direct approach is another limitation as it linearly increases with the increase in target sequence length [16].

Early studies on multi-step prediction using direct approach are done with auto-regression model for forecasting and estimating indicators in economic area [17]. In various studies, results of direct approach are comparatively analyzed against recursive approach using nonlinear data [18], and it is concluded that the direct approach shows better results for long target sequences in terms of prediction error, parameter estimation error, and efficient finite data samples [19]. A study for vehicular route prediction defines mobility patterns as consecutive series of road segment selections. It uses multiple Variable-order Markov Models (VMMs) to mine mobility patterns of different road segments from the real taxi GPS trace data, and results show only 40% of next road segments are correctly predictable for a taxi [20]. The use of Artificial Neural Network for a learning model overcomes the limitation of low accuracy in nonlinear data [21]. The computational and time costs rapidly increase in this approach due to the use of Recurrent Neural Network (RNN) which learns the temporal dependencies of time-series data. In general, direct multi-step prediction approach shows low overall low accuracy as it ignores temporal dependencies in data, and is not scalable because of linearly increasing computation cost. Due to these limitations, direct approach-based models are not suitable for time-sensitive wireless networks, and this paper focuses only on recursive and multi-output approaches for mobile path prediction.

B. Recursive Approach for Multi-Step Prediction

The recursive approach achieves multi-step prediction through recursive iterations of a single model. In each iteration, the learning model predicts single element of target sequence that is used in the next iteration to predict the next element. Generic nature of this approach allows various ML/DL models to be designed in a recursive manner which enables use of new sophisticated models for multi-step prediction. Limitation of recursive approach is that the learning model uses predicted output in each iteration to update the parameters, and this restricts the optimization of parameters for multi-step prediction [22]. Furthermore, the error between predicted and target values in each iteration are accumulated due to recursive use of the predicted values regardless of their accuracy [23]. With consistent high prediction accuracy in each iteration, error accumulation can be significantly mitigated [24].

An earlier study on mobility pattern prediction in cellular networks utilizes ML based model iteratively to calculate HMM that predicts randomly generated user path in simulated environment [25]. The results show that the model achieves lower than 80% accuracy in predicted user path with a major limitation of high computational complexity. A similar study uses Semi-Markov model with recursion to predict user paths with respect to sensing tasks (Point of Interests) based on GPS coordinates data [26]. Later predicted paths are reduced to a limited area mainly containing the PoI sensing tasks. Recent studies on recursive approach have intuitively leverage from high performance capability of various models to overcome the error accumulation limitation. To this end, Support Vector Machine (SVM) is proposed to reduce the stepwise prediction errors where its hyper-parameters are tuned for multi-step prediction [27]. More recently, DL models are forged to fit the recursive approach for multi-step prediction, and a study exploits RNN to predict flow patterns of road traffic [28]. Similarly, LSTM is recursively used to improve the detection of abnormal diagnostic signals from electronic health records [29]. To reduce the error accumulation and improve prediction performance, hybrid direct-recursive approach has been adopted in [30] which is an extension of recursive approach. It uses as many models as there are elements in the target output sequence (*i.e.*, direct approach), where output of a model is used as an input for the next (*i.e.*, recursive approach) Although this approach improves the results, but has excessive computational cost and limited scalability.

Recursive approach with DL models has been used for the path prediction as well, and the time-series data used for this purpose mostly consists of GPS coordinates or cell identifiers in mobile networks. A study predicts the path of a vessel by using GPS coordinates data that is augmented with automatic identification system messages [31]. Another GPS-based dataset [32] is used by recursive LSTM model for multi-step prediction through periodic pattern mining [33]. The accumulated error problem of recursive approach persists in these studies due to limited accuracy of the prediction models which serves as a major limitation for this approach. For this reason, recursive approach is often utilized to generate sequences in heterogeneous model structures [15], [34]. This paper tackles the error accumulation limitation by not only increasing the prediction accuracy of the model but integrating ground-truth induced model training that eliminates the use of predicted outputs in the learning process.

C. Multi-Output Approach for Multi-Step Prediction

The learning model in multi-output approach maps input values to simultaneous multiple output values. This is contrary to previous two approaches where single-step prediction models are extended for multi-step prediction. Hence, the learning models used for multi-output approach are specifically designed for the purpose of giving multiple outputs from a single input sequence [15], and they fully exploit the temporal dependencies of time-series data to achieve higher prediction accuracy [35]. However, learning models based on this approach require excessive datasets, have high computational and time costs, and have low flexibility (e.g., generalization) due to increased complexity of the computations and structure.

The error accumulation problem is less severe in multi-output approach-based models, and this enables long-term predictions with better accuracy. A study uses a multi-output SVM model to perform multi-task learning for long-term forecasting of air quality [36]. Another study improves the results of multi-step prediction in wireless sensor networks by exploiting both spatial and temporal features of data through combination of LSTM and Convolutional Neural Network [37]. However, Seq2Seq [38], [39] is the most prevalent solution for multi-output prediction and is widely used in various domains [40], [41] due to its capability of predicting output sequences of variable lengths with high accuracy.

The earlier efforts for user path prediction through multi-output approach involve pattern mining by using temporal and sequential characteristics of user movement data. Pattern recognition is used to determine patterns of user movements from the historical movement data, and they are used to predict the user path [42]. However, the pattern recognition-based models have low generalization capability, and they suffer from nonlinear characteristics of user mobility. Recent studies have focused on utilizing DL-based multi-output models for user mobile path prediction, such as encoder-decoder model that is built upon Seq2Seq framework [15], [34] and multi-modal recurrent framework with multi-output approach [43]. A recent study utilizes encoder-decoder model architecture with Gated Recurrent Unit (GRU) to predict mobile trajectories [44]. It enhances the prediction accuracy by implementing the attention block in the encoder at the cost of elevated computation. Another work exploits the encoder-decoder architecture in Transformer network model to predict users mobility which is used as an input for maximizing the coverage of Ariel Base Stations by optimizing their positioning [45]. However, performance of these models in terms of prediction error and accuracy is still not sufficiently reliable and consistent to warrant their inclusion in live wireless networks. Additionally these studies ignore the computational complexity aspect of prediction models which is crucial factor in making DL based prediction models viable for live mobile networks.

III. Mobile Path Prediction Model

This section presents preliminary techniques for proposed models, collected dataset with preprocessing algorithm, and two proposed models based on DL for mobile path prediction. In Section III-A, we provide preliminary discussion on LSTM that is used for both of proposed models. The dataset collection and preprocessing algorithm are explained in Section III-B. The proposed SFI model based on recursive approach is described in Section III-C, and ED model based on multi-output approach is described in Section III-D.

A. Preliminary

Temporal dependency is one of the characteristics in time-series data where values are in chronological sequence. The prediction models for time-series data require the learning unit which can reflect the temporal dependency to predict the output. To learn sequential features from the time-series data, RNN [46] is proposed that computes inputs recursively through connected structure of equivalent cells. However, RNN has limitations in learning long sequences due to the gradient vanishing problem [47] that restricts the model to establish dependency relationship between the values in earlier steps and latest steps (*i.e.*, long-term dependency). To tackle long-term dependency problem, LSTM is proposed that adds new state vector namely cell state. The cell of LSTM in step t requires three input values that are input data x_t, cell state c_{t-1} and hidden state h_{t-1}, where c_{t-1} and h_{t-1} are computation results of the cell in previous step, as shown in Fig. 2. These inputs are used by three gates that are a forget gate f_t, an input gate i_t, and an output gate o_t for computation. Forget gate f_t controls the level of influence from cell state of previous step to new cell sate by removing less-related features. Input gate i_t decides how to incorporate new data for updating cell state, while output gate o_t resolves how updated cell state to be forwarded as hidden state that includes the features to be used in subsequent cells.

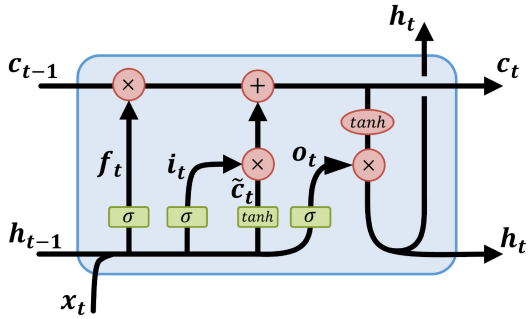


Fig. 2. Long Short-Term Memory cell architecture [48].

Forget gate f_t takes input data x_t and hidden state h_{t-1} as inputs to the sigmoid with their corresponding weights, and the output is multiplied with previous cell state c_{t-1} to determine the impact of c_{t-1} as the scale of sigmoid ranges from 0 to 1 (1). Input gate i_t uses activation function and inputs similar to f_t with its corresponding weights (2), while hyperbolic tangent *tanh* is used to amplify features of these inputs and produce $\tilde{c}_t$ as an output (3). The calculated i_t and $\tilde{c}_t$ are multiplied and the resultant features are added with $f_t \times c_{t-1}$ to give new cell state c_t (4). Output gate o_t uses similar inputs and activation function as i_t, however, o_t utilizes scale of sigmoid function to determine level of cell state effect to the hidden state h_t (5). h_t is calculated by multiplication of o_t and output of *tanh* whose input is c_t, where purpose of *tanh* is to magnify information in c_t that is required for prediction in next step (6). The composition of three gates in LSTM and their relations are shown in Fig. 2, where $\times$ stands for Hadamard product. In (1)-(6), W and b stand for weights and bias of subscripts accordingly, and the training of weights is done through the Backpropagation Through Time (BPTT) algorithm [49].

$$f_t = \sigma\big(W_{xf}\, x_t + W_{hf}\, h_{t-1} + b_f\big) \tag{1}$$
$$i_t = \sigma\big(W_{xi} x_t + W_{hi} h_{t-1} + b_i\big) \tag{2}$$
$$\tilde{c}_t = tanh\big(W_{xc} x_t + W_{hc} h_{t-1} + b_c\big) \tag{3}$$
$$c_t = f_t \times c_{t-1} + i_t \times \tilde{c}_t \tag{4}$$
$$o_t = \sigma\big(W_{xo} x_t + W_{ho} h_{t-1} + b_o\big) \tag{5}$$
$$h_t = o_t \times tanh(c_t) \tag{6}$$

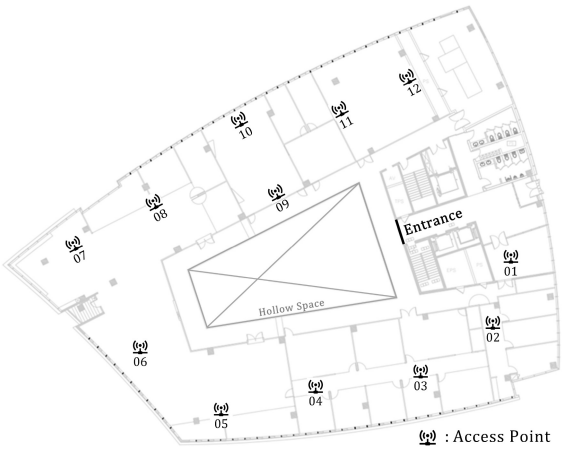


Fig. 3. Pangyo ICT Research Center floor plan with highlighted 12 APs.

B. Data Collection and Preprocessing

The performance of DL models in time-series learning is highly dependent on the characteristics of data. Regardless of the actual result values, it is more important to establish superiority of a DL model over others independent of the data, environment, and scenario. To achieve this, we have processed two raw logs, one from the campus wireless network with 12 APs and other from 62 BSs of one of the 5G mobile network operators in Korea, to create Campus Mobile-path Dataset (CMD) and Operator Mobile-path Dataset (OMD), respectively. It is worth noting that the proposed SFI and ED models can be used for multi-step ahead prediction over any time-series data with appropriate changes in the loss functions as per data type. Moreover, lack of auxiliary information such as PoA location/range and signal strengths in open-source GPS datasets make them unsuitable, as focus of this study is to find user path in terms of PoAs to facilitate proactive mobility and resource management.

The path of user mobile can be defined as a sequence of identifiers (IDs) that represent the Base Stations (BSs) to whom mobile has been connected in the chronological order. To generate mobile-path dataset, we collect handover logs [13] of the mobiles from wireless network of Pangyo ICT Research Campus [50] located in Pangyo, Korea. As shown in Fig. 3, the research center is an area of approximately $1587.74m^2$ including highlighted 12 Access Points (APs) with an average of 70 people/users connecting to wireless network per day. The user movements cause handover between the APs, and the logs are collected by syslog program in AP controller for two months. The total number of logs is 171, 141 and each log entry consists of 3-tuple (mobile ID, Source AP, and Target AP). The IDs are distinguished through MAC addresses of the devices, and 12 APs are recognized by integers from 0 to 11.

The log entries are manipulated to create a sequence that represents a path. A dataset of path sequences is created through following three steps of preprocessing: classification of mobile ID, concatenation, and normalization for sequence length. The log entries of a mobile are classified using its ID, which represents sequence of the movement since the handover logs are collected in the order of their occurrences. The classified log entries are concatenated to generate sequences when the target AP of x^{th} log equals to the source AP of $x+1^{th}$ log. For example, the sequence $\langle 1, 3, 5, 4\rangle$ is generated when three log entries for mobile identifier *i* are given in the order (*i*, 1, 3), (*i*, 3, 5), and (*i*, 5, 4). After this process, we normalize the lengths of all the created sequences to unify them based on the target length of combined input and output sequences for model training and testing. The sequences that are shorter than the target length is removed, and the longer sequences are normalized to target length through splitting with shifting, and this makes the models agnostic to network scaling. For example, when the sequences $\langle s_1, s_2\rangle$ and $\langle s_1, s_2, \ldots, s_5\rangle$, where s_j is AP identifier at index *j*, are given with the target length 3, the normalization process results in removal of s_1, s_2 and the longer sequence is converted to multiple subsequences $\langle s_1, s_2, s_3\rangle, \langle s_2, s_3, s_4\rangle, \langle s_3, s_4, s_5\rangle$. At the end of these three steps, we shuffle the order of created sequences to prevent biased batch that limits training performance of the models due to similar trend of subsequences from same mobile [8]. Algorithm 1 describes the path sequence manipulation process.

The elements s_j in the sequences are given as input for the models, and they are computed in the LSTM cells. All the sequence elements indicate AP ID in scale of 0 to 11, however, two proposed models require different input data formats. On this account, the elements are transformed into different shapes as per the SFI and ED models requirements through the processes of (1) and (2), respectively. (1) The element for SFI model is converted to [1, 12] and divided by the number of APs to rescale it in the range of (0, 1]. The rational to dismiss the zero from AP representation is because its usage causes the convergence of intermediate values to zero. (2) The element in ED model dataset is tokenized through one-hot encoding to map multiple elements of input and output effectively. Consequently, the element is represented in one-hot vector of 12 dimensions that represent 12 APs. For example, AP ID 3 is converted into vector [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0] and 5 into vector [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]. This representation of elements is adapted for the comparison models, GRU-Attention (GRU-ATTN) model [44], and Transformer Network (TN) model [45].

The raw data for OMD are acquired from a 5G network operator in South Korea. The raw data in OMD consists of 110 million handover logs collected over 15 days from an Access and Mobility Function (AMF) in 5G covering 62 PoAs and hosting on average 3.2 million subscribers per day. To ensure the subscribers privacy, the collected logs comply legal guidelines for personal information anonymization as per Korean law. A raw log entry is a 4-tuple of Device ID, source BS ID, destination BS ID, and timestamp. The aforementioned mobile path sequences creation process for CMD is similarly applied on OMD raw logs to create mobility sequences.

C. Recursive Approach Based Model

The proposed SFI model uses recursive approach to predict the mobile path of target length *q* from a given historical path of length *p*. This model is an extension of our previous work on single-step prediction [8] that predicts next PoA by using

Algorithm 1: Path Sequence Manipulation

Input: The handover logs G, sequence length l
Output: Dataset D of path sequences of length l

```
// Step 1: dividing handover logs based on mobile IDs
for each d ∈ G do
    // d is a 3-tuple of (uID, sourceAP, targetAP)
    i ← d.uID
    U ← U ∪ {i} // set of mobile IDs
    Insert d into G_i, the list of handover logs for mobile ID i
// Step 2: creating path sequences from the handover logs
S ← [ ] // list of path sequences
for each i ∈ U do
    prev ← −1, seq ← ⟨ ⟩
    for each d ∈ G_i do
        if d.sourceAP ≠ prev then
            if seq ≠ ⟨ ⟩ then
                Insert seq into S
                seq ← ⟨ ⟩
            // make a new sequence with source and target APs
            seq ← ⟨d.sourceAP⟩‖⟨d.targetAP⟩
        else
            // append target AP into the current sequence
            seq ← seq‖⟨d.targetAP⟩
        prev ← d.targetAP
    if seq ≠ ⟨ ⟩ then
        Insert seq into S
// Step 3: normalizing path sequences to length l
D ← [ ] // list of normalized path sequences
for each seq ∈ S do
    if len(seq) ≥ l then
        j ← 0
        while j ≤ len(seq) − l do
            subseq ← ⟨ ⟩
            for k ← 0 to l − 1 do
                subseq ← subseq‖⟨seq[j + k]⟩
            Insert subseq into D
            j ← j + 1
Shuffle the order of sequences in D
```

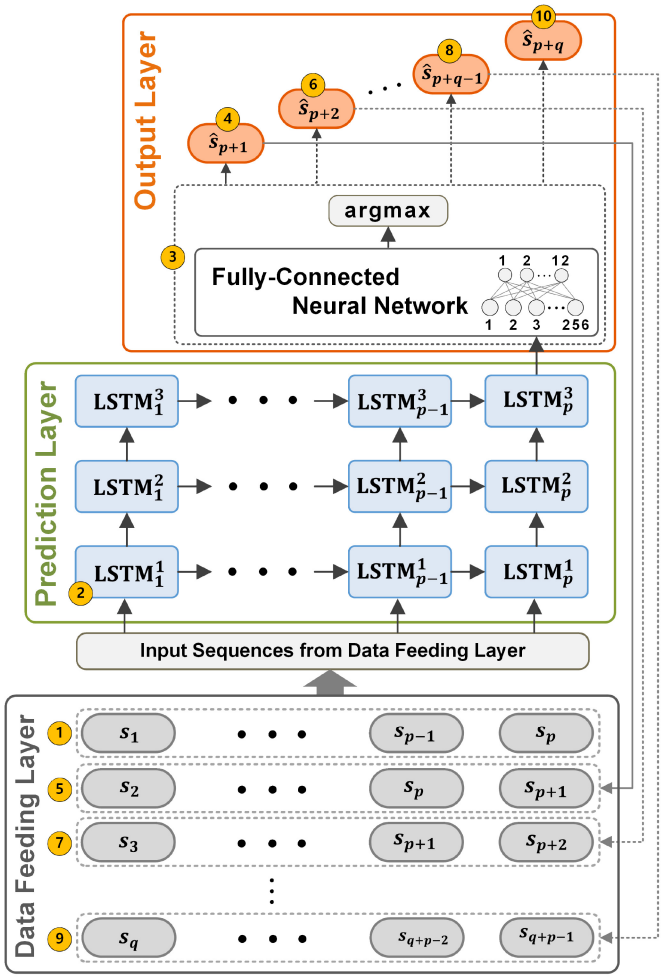


Fig. 4. The architecture and operational flow of the proposed SFI model.

stacked LSTM. As shown in Fig. 4, the single-step prediction model is evolved into a three-layered architecture for SFI model, which includes Data Feeding Layer, Prediction Layer, and Output Layer. As an input in SFI model, the sequence $\langle s_1, s_2, \ldots, s_p \rangle$ ① from Data Feeding Layer is inserted into LSTM cells in the first sublayer of Prediction Layer ②. The computation result through three sublayers of LSTM cells is generated as 256-dimensional vector. The vector is fed to the Fully Connected (FC) neural network ③ in the Output Layer, and it is converted to 12-dimensional vector. The index of maximum value in the vector is selected to represent predicted next AP ID $\hat{s}_{p+1}$ ④ via argmax function, and $\hat{s}_{p+1}$ is appended at the end of input sequence to generate new sequence ⑤ for predicting next step $\hat{s}_{p+2}$. This process continues iteratively $q - 1$ times (⑥ - ⑩), and subsequent paragraphs describe the detailed operation of each layer in the SFI model architecture. Furthermore, the operational flowchart of SFI model is illustrated in the Appendix.

The Data Feeding Layer takes as sequence $\langle s_1, s_2, \ldots, s_p \rangle$ of length p as an initial input to predict $\hat{s}_{p+1}$, and uses shift-and-join method to create new inputs in next iterations for predicting target path sequence $\hat{s}_{p+2}, \hat{s}_{p+3}, \ldots, \hat{s}_{p+q}$. The shift-and-join method performs one step left shift on the input sequence of previous iteration and appends the latest predicted result at the end. These sequences reconstruction using shift-and-join method and prediction results are repeated $q - 1$ times to output predicted mobile path of length q. For example, an initial input sequence $\langle s_1, s_2, \ldots, s_p \rangle$ is passed to Prediction Layer without any process as shown in Fig. 5. The shift-and-join method creates a new sequence which is used to predict $\hat{s}_{p+2}$ through upper layers, and same processes continue in $q - 1$ times. As shown in Fig. 5, the last predicted result $\hat{s}_{p+q}$ is not used for the reconstruction process as the sequence of target length q is completely predicted.

The LSTM sublayers of the Prediction Layer in SFI model are trained by updating computational parameters through loss function in each iteration. However, the parameters can be updated in the wrong direction to predict targets (*i.e.*, ground-truth) when incorrect prediction results are used in shift-and-join method to create new input sequences. Moreover, the loss is continuously increased in subsequent iterations since the appended values are shifted and used in all the next iterations,

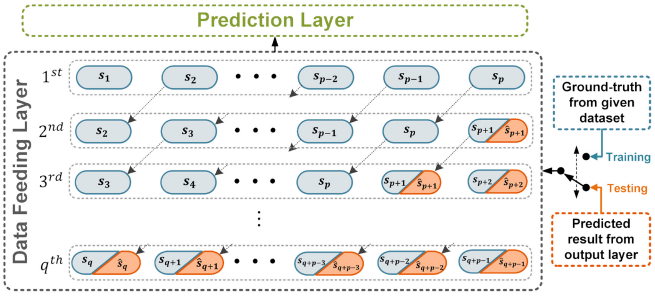


Fig. 5. Data feeding layer in SFI model with transformation of single sequence into $q-1$ sequences.

and this causes high time-cost to converge parameters. To avoid this problem, discounted significance of accumulation mechanism is used for assigned inputs. In this mechanism during training, the ground-truth value at each step of output sequence is not only used for calculating loss but is also used as an input in the data feeding layer for the next step prediction instead of the predicted value. The different values for sequence reconstruction in the training and testing processes are depicted in Fig. 5, where the predicted values are highlighted in orange with hat-symbols and ground-truth in blue. The switch in the right of Fig. 5 indicates that predicted and ground-truth values are used for testing and training, respectively, and this transition happens only once in an experiment.

The Prediction Layer of the SFI model employs three sublayers of $LSTM_w^v(v = 1, 2, 3 \ and \ w = 1, 2, \ldots, p)$ cells, where v is the layer number and w indicates the step. The sublayers include p LSTM cells to compute input sequence of p elements which are passed from the Data Feeding Layer. The first element of the input sequence is computed by $LSTM_1^1$ gates to 256 dimensional vectors of cell state and hidden state as described in Section III-A. The $LSTM_2^1$ computes new cell state and hidden state using second element of input sequence and the result of previous cell, and this process repeats for $p-1$ times. The process of the first sublayer is repeated similarly in the upper layers which are stacked to increase the number of cells and enables the model to learn complicated temporal features through storing values in larger space. At the end of the process, hidden state computed by $LSTM_p^3$ contains key temporal features of the sequence and is passed to the Output Layer for producing the predicted result.

The Output Layer consists of a FC neural network and an argmax function similar to structure of a typical classification model that outputs by selecting from multiple options based on their correctness probability. As shown in Fig. 4, FC neural network converts the received 256-dimensional vector to 12-dimensional vector through its 12 neurons with Softmax activation function that normalizes all elements to fit in [0, 1] with their sum as 1. This process results 12-dimensional vector representing the probability of each AP, and AP ID of next predicted step is selected from the vector through the argmax function that returns the vector index of the maximum value. During the testing phase, the predicted AP ID in this iteration is passed to the Data Feeding Layer where the input sequence for next iteration is manipulated.

The training process uses the predicted AP ID in each iteration to calculate the loss by comparing it with the ground-truth value. Loss calculation is done through categorical cross-entropy loss function that is widely used for typical classification models. The 12-dimensional score (*i.e.*, probability) vector from softmax function in the Output Layer and ground-truth in one-hot vector format are used for calculating loss, which is utilized by an optimizer that handles parameter update. Adam optimizer that combines Stochastic Gradient Descent with Momentum and RMSProp updates the trainable parameters to minimize the loss. This process continues q times for one sequence, and this repeats for each sequence in the given training dataset.

D. Multi-Output Approach Based Model

The multi-output approach-based ED model predicts a mobile path sequence of length q from historical path sequence of length p through Encoder network and Decoder network, which are illustrated as left and right blocks in the Fig. 6, respectively. The Encoder network processes the input sequence through a FC neural network and three sublayers of LSTM cells to create Encoder state that contains compressed representation of all the elements. The Encoder state is passed to the Decoder network that simultaneously predicts q elements of target sequence by using two FC neural networks and three sublayers of LSTM cells. In the subsequent paragraphs, we first describe the overall functioning of the ED model and then provide the detailed account of the Encoder network and Decoder network operations.

The Encoder network takes the elements of given input sequence $\langle s_1, s_2, \ldots, s_p \rangle$ ① that are individually transformed to vectors through the FC neural network ②. The vectors are sequentially computed by $LSTM_w^v(v = 1, 2, 3, and w = 1, 2, \ldots, p)$ cells ③, and the results of cells in the last step $LSTM_p^v(v = 1, 2, 3)$ represent the Encoder state and are conveyed to the Decoder network ④. The cells $LSTM_w^v(v = 1, 2, 3, and w = 1, 2, \ldots, q)$ ⑤ in the Decoder network are initialized using the Encoder state, and the Start of Sequence (SoS) vector ⑥ is passed to $LSTM_1^1$ through the FC neural network ⑦. The cell computations are proceeded from the $LSTM_1^1$ ⑧ to cells in upper layers $LSTM_1^v(v = 2, 3)$, and their result is utilized by the FC neural network and the argmax ⑨ to output AP ID $\hat{s}_{p+1}$ ⑩ (*i.e.*, first element of the target sequence). $\hat{s}_{p+1}$ is used as input ⑪ for cell computations in next step $p+2$, and this process is repeated $q-1$ times to generate the predicted mobile path of length q (⑫ - ⑯). Furthermore, the operational flowchart of ED model is illustrated in the Appendix.

The FC neural network in the Encoder network takes sequence of length p as input and converts its elements into 256-dimensional vectors for the cell computation through 256 neurons of the neural network. As shown in the left block of the Fig. 6, $LSTM_1^1$ ③ takes input vector to compute the cell state and hidden state which are passed onto $LSTM_2^1$ on the right and only hidden state is passed above to $LSTM_1^2$. This sequential computation continues from left to right $(w = 1, 2, \ldots, p)$ and bottom to top ($v = 1, 2, 3$). The

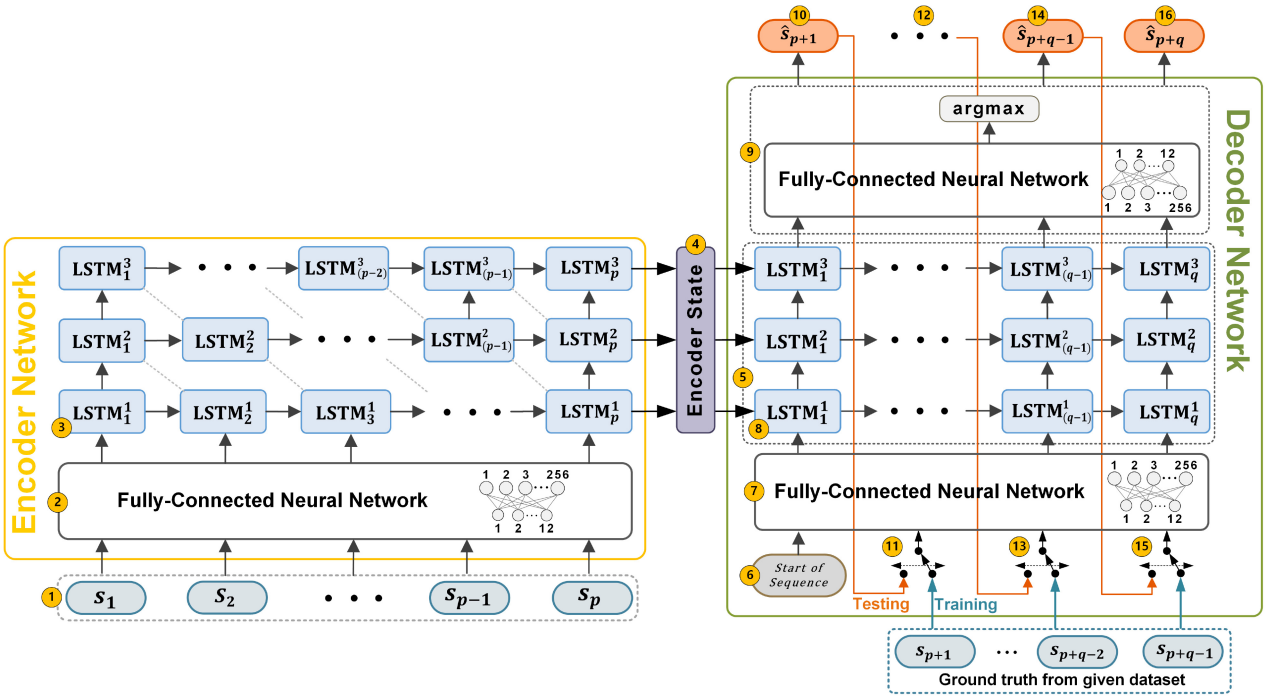


Fig. 6. The architecture and operational flow of the proposed ED model.

hidden and cell states vectors of the p^{th} cell in each sublayer $LSTM_p^v(v = 1, 2, 3)$ defines the Encoder state which contains extracted temporal features from entire input sequence. These vectors are passed to the $LSTM_1^v(v = 1, 2, 3)$ of the Decoder network for state initialization, and this enables ED model to operate jointly for the processing of input and output sequence.

The Decoder network consists of two FC neural networks and three sublayers of LSTM cells, where the computation proceeds from bottom to top $LSTM_1^v(v = 1, 2, 3)$ and carry-ahead method then moves the result one step to the right. In particular, FC neural network ⑨ takes the hidden state of $LSTM_1^3$ and outputs the first element (*i.e.*, carry) of the output sequence $\hat{s}_{p+1}$⑩ that is moved ahead as an input to $LSTM_2^1$ for predicting $\hat{s}_{p+2}$. This carry-ahead operation continues for additional $q - 1$ times to generate predicted sequence of length q. However, this operation differs from the recursive intake process in SFI model, as the operation does not reshape the input sequence and only impact remaining cells in the network. Additionally, it is worth mentioning that the input for $LSTM_1^1$ is a 12-dimensional SoS vector consisting of random values that are created once at the time of model initiation and then used for each iteration after being transformed to 256-dimension by FC neural network ⑦.

The target sequence generation of length q from the Decoder network through recursive operations improves with the teacher forcing method in training, where the ground truth values of the target sequence are used instead of the predicted values. This is similar to the sequence creation operation in SFI model, and is depicted in right block of Fig. 6 as switches ⑪, ⑬, and ⑮ which provide ground-truth and predicted values to the Decoder network in training and testing processes, respectively. Another crucial aspect of ED model is its training of both the Encoder and the Decoder networks by using the multiple outputs of the Decoder network. For this, loss value for each predicted element is individually calculated through categorical cross-entropy loss function, and this value uses Adam optimizer to update the trainable parameters in all the preceding steps including itself. For example, calculated loss value of last predicted element $\hat{s}_{p+q}$ is utilized to update parameters from $p + q$ to 1, whereas loss value of last predicted element $\hat{s}_{p+1}$ is utilized to update parameters from $p + 1$ to 1.

The training computation time of both SFI and ED models is insignificant as training is done offline, however, time required for predicting a target sequence in online deployment is important for realtime systems [51]. To this end, we quantify the computation time of both the models for single target sequence inference. Computation time quantification can be done through various criteria [52], however, this paper utilizes the operational flow of the models in conjunction with number of LSTM cells and layers to derive the computation time of single target sequence inference. SFI model and encoder network in ED model consists of 3 layers with p LSTM cells in each, whereas, ED model decoder network has 3 layers with q LSTM cells in each. As shown in the Fig. 6, the dotted grey lines in the encoder network represent simultaneous computations in r time due to parallelized computation property of LSTM, where r is max unit computation time for a single or simultaneous LSTM cells. The p computations from $LSTM_1^1$ to $LSTM_p^1$ are completed in $p \cdot r$ time, and additional $(3-1) \cdot r$ time is required for $LSTM_{p-1}^3$, $LSTM_p^2$, and $LSTM_p^3$ cells computation. Hence, it takes $(p+2) \cdot r$ time to compute LSTM with p cells and 3 layers. From this, the computation time of LSTM with x cells and y layers is generalized into $(x + (y-1)) \cdot r$. As SFI model repeats q times to predict

TABLE I
TRAINING PARAMETERS FOR SFI AND ED MODELS

Parameters	Descriptions	Values
Batch size	The number of sequences to work through before updating the model parameters (i.e., weights)	100
Epoch	The number of times that the learning algorithm will work through the entire training dataset	300 (110)
Learning rate	How much to change a model in response to an estimated error each time the model weights are updated	0.01
Dimension	The dimensionality of a hidden layers in LSTM	256
Input sequence Length	The number of elements in input sequence	14-22 (70-110)
Target sequence Length	The number of elements in target(i.e., output) sequence	3, 5, 7

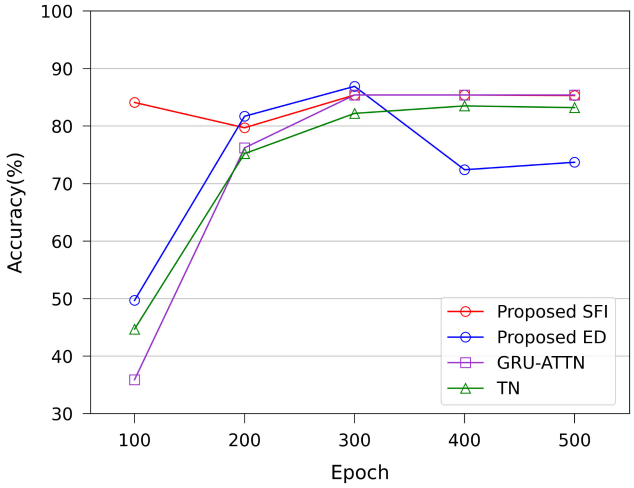


Fig. 7. The accuracy of four models for varying epochs with CMD.

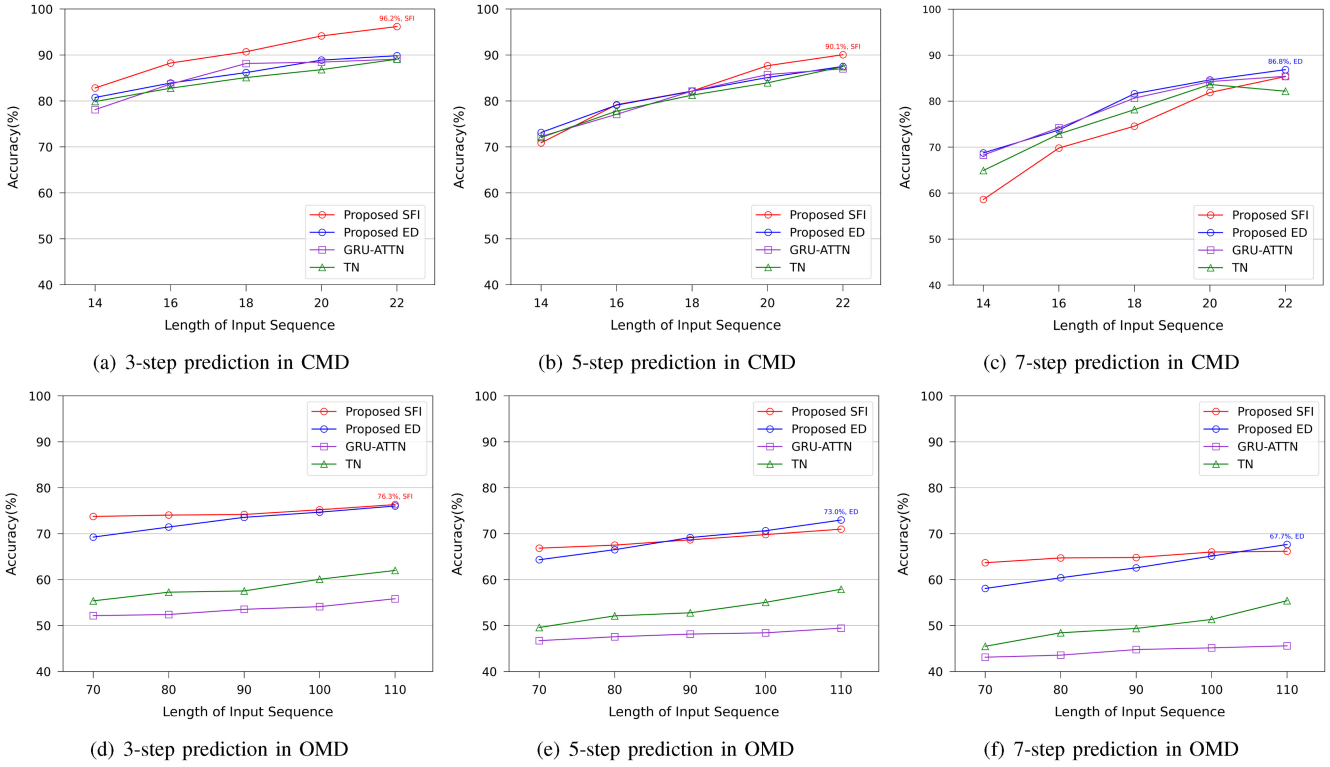
sequence of length q, the computation time of the model for single sequence prediction is $q \cdot (p+2) \cdot r$. For ED model, the computation of the encoder network is $(p+2) \cdot r$ that is similar to SFI model. Whereas decoder network sequentially computes three LSTM cells (i.e., one in each layer) to predict an output which is carried-ahead to the next step, and this continues for q times. Hence, decoder network computation time is $q \cdot (3r)$, and for ED model it is $(p+2+3q) \cdot r$.

IV. RESULTS AND ANALYSES

The two proposed models are implemented using Keras and CUDA libraries on the hardware environment consisting of Intel i7 CPU, 64GB RAM, and RTX 2080 GPU. The dataset is divided into training and testing data with 70:30 ratio, to provide comprehensive performance evaluation while maintaining enough data for training [8], [53]. All four models are trained for 300 epochs (110 epochs for OMD) using the training data with batch size of 100. The testing data is used to conduct experiments on both models, where each presented result is an average of 30 experiments. The maximum number of steps in the predicted target sequence are 7, because maximum length of sequences in CMD is 29 and with input sequence length 22 models can only be trained for predicting next 7 steps. As the number of BSs in OMD are 5 times more than the number of APs in CMD, we have increased input sequence length 5 times to compensate for increased network scale and mobile path diversity. Detailed parameters and their values are shown in Table I for replication of experiments and results.

Hyperparameters play an important role in model performance, among which epoch is the most significant. The epoch value refers to training iterations required for converging model to optimal value, hence, epoch value is a function of trainable parameters in the model. A low epoch value results in partially trained model (*i.e.*, underfitting), and a high epoch value causes a model to over train (*i.e.*, overfitting) which is also detrimental for models' performance. To determine the best epoch value for the evaluation of the proposed and target models, experiments with increasing epoch values are conducted for 7 step prediction accuracy of all the models using CMD, and the results are shown in Fig. 7. The highest number of trainable parameters are in ED model for 7-step prediction, and the optimal epoch value in Fig. 7 for 7-steps confirms that model also converges to optimal value for smaller-step as well. For higher-step predictions, epoch values need to increase due to more trainable parameters in the model. The abnormal behavior of SFI accuracy from 100 to 200 epochs can be explained by local and global optima. The local optima and shows higher accuracy which later drops at 200 epochs after coming out of local optima. At 300 epochs the model converge to global optima as its performance remains consistent for 400 and 500 epochs. Based on these results, epoch is set as 300 for the remaining experiments with CMD.

Multi-step ahead prediction enables mobile operators to not only proactively manage the user mobility but also assign, reassign, or scale the network, and compute resources in MECs proactively to enhance user experience. Consistent accuracy of predictions plays a significant role in this, as wrong prediction degrades service quality and causes overhead in decommissioning proactively assigned mobility resources and reassigning them reactively. To this end, outright accuracy results of three, five, and seven steps ahead predictions for SFI model and ED model are depicted in Fig. 8 as a function of input sequence length, and compared against two state-of-the-art models GRU-ATTN [44], and TN [45]. For the comprehensive analyses, the results of CMD and OMD are separately illustrated in Figs. 8 (a) to (c) and (d) to (f), respectively. Here, outright accuracy denotes that each step in the output sequence is correctly predicted. In Figs. 8 (a) and (d) for three steps ahead predictions, SFI model shows highest accuracy of 96.2% and 76.3% with 22 and 110 input sequence lengths, and outperforms other models by maximum 7% for CMD and 21% for OMD. In case of CMD, the gap reduces to 3% as predicted output sequence length increases to five, and this performance gap reverses for OMD result that ED shows 2% higher than SFI. For seven steps ahead predictions, ED model outperforms SFI model by 2% in Fig. 8 (c) for CMD, and this continues in Fig. 8 (f) for OMD where ED performs 1.5% better than SFI. The results in Fig. 8 indicate that SFI has a definite edge over ED for output sequences of smaller length like three or five, whereas ED model performs better in terms



(a) 3-step prediction in CMD (b) 5-step prediction in CMD (c) 7-step prediction in CMD

(d) 3-step prediction in OMD (e) 5-step prediction in OMD (f) 7-step prediction in OMD

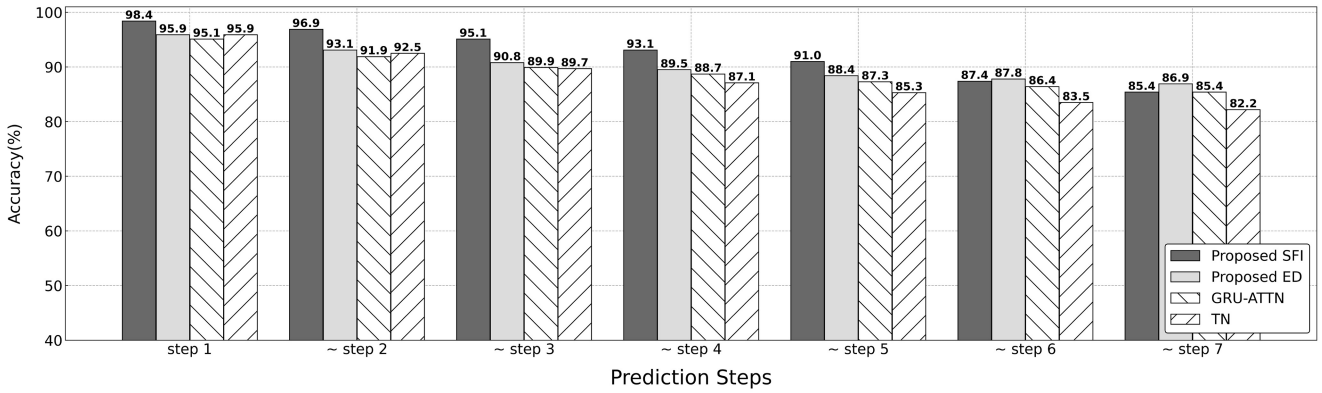
Fig. 8. The overall accuracy of four models in various prediction steps with Campus Mobile-path Dataset (CMD) and Operator Mobile-path Dataset (OMD).

of accuracy for output sequence of length seven. Moreover, the performance drop of GRU-ATTN and TN models for OMD suggests their unsuitability for mobile path prediction.

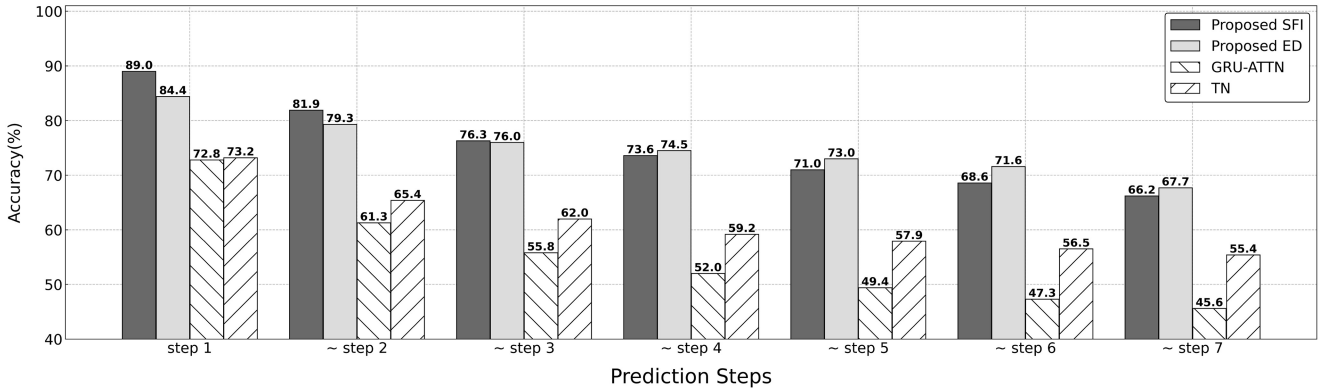
The results in Figs. 8 (a) to (c) show a direct correlation between accuracy increment and increasing input length, as the overall accuracy of four models improves with the increasing sequence length up to 22 steps. From this upward trend it can be estimated that the accuracy will further improve with increment in input sequence length, however, our experiments on the CMD show a drop in prediction accuracy for 7-step target length when the input sequences are length of 24 and above. This is presumed to be caused by under-fitting that is occurred due to insufficient dataset for training. The required length of the sequences for the training is the sum of input and target sequence lengths $p + q$, and sequences that are shorter than this length are not used in the training of the models. Hence, with input length 24 and target length 7 the required length of sequences for training is 31, and in the wireless network of 12 APs the movement of users is limited that generate insufficient long sequences in dataset. Moreover, this result provides an approximate upper bound on the accuracy of multi-step prediction as a function of input sequence length in small wireless networks.

The similar trend with CMD results is confirmed in OMD results of Fig. 8 (d) to (f) that the overall accuracy of four models tends to improve with the increasing sequence length up to 110 steps. The maximum accuracy in OMD is 76.3% by SFI for three steps ahead prediction, which is 19.9% lower than CMD result. This decrease is attributed to much larger scale of mobile network where OMD is collected and diverse mobility patterns of users with higher entropy. As similar with CMD results, SFI outperforms all other models for 3-step prediction, however, ED dominants from 5 to 7 step predictions with the sequence length 110 steps. The maximum performance gap between the proposed and comparison models significantly increases to 21.17% from 7.09% in CMD results. This is because the comparison models are based on attention mechanisms which are used for natural language tasks to improve a model performance by learning correlations of the words in input and output sentences. In the mobile path prediction tasks, attention over PoA (*e.g.*, AP or BS) IDs does not provide the necessary correlation for improvement since PoA IDs themselves are uncorrelated and it is temporal/sequential correlation that is required.

The predicted output sequence defines ordered future PoAs of the user and provides a sense of his/her mobile path. A wrong prediction at any step of the predicted sequence is unacceptable for proactive mobility management as it is handled on per hop basis. However, MEC assignment in 5G and allocation of resources are done based on general notion of the user mobile path, where low prediction accuracy of one or two steps in a long output sequence is tolerable. This warrants a stepwise analysis of prediction models for seven steps, as it is the longest prediction steps in our experiments. Here, correct predictions at each step are only accounted if previous step is correctly predicted as well. The results of four prediction models for CMD in Fig. 9 (a) show a consistent and gradual decrease in prediction accuracy from first to last step when the input sequence length is 22. The average decrease in SFI and TN is 2.17% and 2.28%, respectively, and for ED and GRU-ATTN it is 1.5% and 1.61%, respectively. The SFI shows higher accuracy than other models till fifth step, after that ED



(a) Campus Mobile-path Dataset (CMD)



(b) Operator Mobile-path Dataset (OMD)

Fig. 9. The stepwise accuracy performance of four models with two datasets.

has 0.4% and 1.5% better results than SFI for sixth and seventh steps, respectively. GRU-ATTN and TN models have lower performance than SFI and ED for all the steps, where TN is better than GRU-ATTN in first two steps and its vice versa for the remaining. The results for OMD in Fig. 9 (b) show the average decrements from first step to last step are increased 0.69% 2.91% compared to CMD results in Fig. 9 (a), and TN achieves the lowest average decrement of 2.97%. Comparing the individual prediction accuracy of any step with the extrapolated result based on decrement trend it can be see that the results are similar. This confirms that the decrement accuracy trend holds and can be used to extrapolate prediction accuracy of steps beyond seven in both the datasets.

The results in Fig. 8 and Fig. 9 establishes that increasing the input sequence length yields better overall prediction accuracy, whereas the prediction accuracy of individual steps decreases as the target sequence length increases. This is because the variance of combinatorial possibilities increases as we move towards tail end of the target sequence. The superior performance of SFI till fifth step in Fig. 9 (a) confirms the effectiveness of discounted significance of accumulated assigned inputs in the training and higher individual step accuracy. However, it has higher decrement slop due to error accumulation, and for that reason ED results become better than SFI for sixth and seventh steps due to lower decrement slop despite having lower accuracy in the first step. This is reaffirmed in results of OMD in Fig. 9 (b) that ED outperforms from the fourth step, as the average decrement of ED and SFI are greatly increased into 2.78% and 3.8%, respectively. The proposed SFI and ED achieve better accuracy than GRU-ATTN and TN for both CMD and OMD. The comparison models depict high performance degradation in OMD as shown in Fig. 9 (b). This is because both of GRU-ATTN and TN start with lower first step accuracy than the proposed models, and GRU-ATTN model has the highest average decrement of 4.5%. Based on the results in Figs. 8 and 9, it can be concluded that SFI and ED comprehensively outperform GRU-ATTN and TN, and for shorter output sequences SFI has clear better performance whereas ED is preferred model for longer output sequences. This is because ED model targets to minimize loss of entire sequence while SFI model aims to curtail loss at individual steps.

The increase in target sequence length (*i.e.*, the number of steps) gradually reduces the prediction accuracy in both the models due to error accumulation. The error accumulation occurs as a result of wrong prediction in step t and recursion mechanism of the models to predict next step $t + 1$. This paper reduces error accumulation in two ways. First, in the training of both models, the accumulation is prevented by the discounted significance accumulation method that provides only ground-truth values. Secondly, SFI and ED models use LSTM as a learning unit which has the best performance when compared with RNN and Gated Recurrent Unit (GRU). This is confirmed by the results shown in the Fig. 10 which are

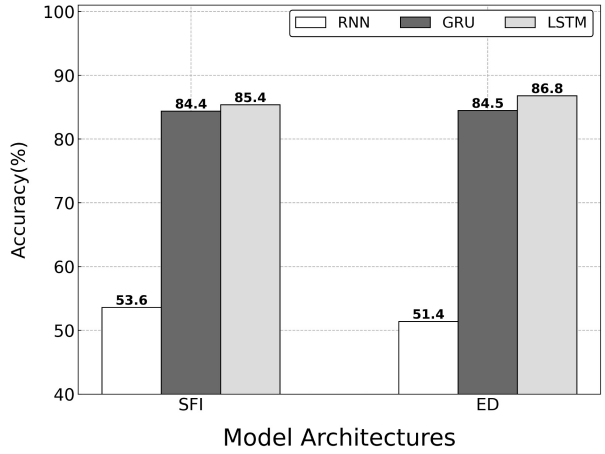


Fig. 10. The accuracy of three different learning units in the architecture of SFI and ED.

TABLE II
AP PRECISIONS OF FOUR MODELS IN CAMPUS MOBILITY DATASET

Models	SFI		ED		GRU-ATTN		TN	
Predictions	Avg	Stdev	Avg	Stdev	Avg	Stdev	Avg	Stdev
AP01	0.98	0.02	0.93	0.02	0.92	0.04	0.95	0.02
AP02	0.89	0.03	0.95	0.02	0.95	0.02	0.93	0.01
AP03	0.96	0.03	0.84	0.04	0.87	0.03	0.82	0.02
AP04	0.98	0.02	0.87	0.13	0.86	0.11	0.97	0.03
AP05	0.94	0.11	0.74	0.10	0.77	0.10	0.97	0.04
AP06	0.94	0.09	0.98	0.02	0.95	0.06	0.91	0.04
AP07	0.98	0.03	0.86	0.25	0.81	0.18	0.83	0.14
AP08	d 0.99	0.02	0.86	0.01	0.88	0.04	0.85	0.13
AP09	0.95	0.06	0.83	0.06	0.92	0.07	1.00	0.00
AP10	0.94	0.02	0.96	0.02	0.97	0.02	0.93	0.04
AP11	1.00	0.00	0.87	0.18	0.76	0.27	0.85	0.04
AP12	0.96	0.02	0.98	0.01	0.99	0.01	0.94	0.06

for input and target sequences lengths 22 and 7, respectively. LSTM enables SFI and ED models to first predict next step $p+1$ with high accuracy which leads to substantial reduction of error accumulation for predicting $p+2$ during testing. Through these two strategies error accumulation problem is effectively subdued in SFI and ED models but cannot be completely mitigated due to inherent characteristics of recursive and multi-output approaches.

The results in Fig. 9 provides a closeness measure of overall predicted output sequence to the ground-truth, and there is a need for further dissection of results to determine how each model performs in predicting q steps for different APs in the network. To this end, the precision of each AP shows consistency of each model correctness when it outputs a particular AP as while predicting q target steps. It is calculated as: $\frac{True\ positives}{True\ positives+false\ positives}$, where true positives are correct model predictions of a particular AP and false positives are model predictions of a particular AP but they are incorrect. In particular, the precision of each AP highlights that how often the model has predicted it incorrectly, where low precision value represent higher number of incorrect predictions. The reason for higher incorrect predictions for certain APs can be understood, when we correlate them with the locations of APs in Fig. 3 and patterns of mobile paths. As the locations of BSs in OMD are unknown, their precision results are not included. For each AP in CMD, the average precision of 3, 5, and 7 steps and their standard deviation are computed with four prediction models, and are shown in Table II. The average value provides an insight that how many incorrect predictions a model has made for an AP, and standard deviation shown the precision consistency of a model for 3, 5, and 7 steps. Although, intuitively the proposed models show higher average precision for each AP due to better accuracy performance when compared to GRU-ATTN and TN, but no clear pattern emerges. A closer look reveals that APs 7, 8, and 9 have higher rate of significantly low average precision values and high standard deviation for at least two models. Based on the layout in Fig. 3, it can be seen that these APs are towards the end of the campus and model predicts them to be the next AP but user may takes a different direction at earlier point, which leads to high incorrect predictions. For the same reason, we can see similarly low precision values for APs 3 and 11. For the remaining APs, all the models show varying results with no particular pattern. Overall, SFI model precision results show more consistent and reliable performance with 97.2% average values above 0.9, whereas in ED model only 61.1% values are above 0.9. On this account, SFI model is more preferable over ED model to be used in mobile networks for mobility and resource management despite its relatively low accuracy for longer target sequences.

We confirm our analytically calculated computation times by empirically investigating training and testing times of both models for 3-step, 5-step, and 7-step predictions. These results are summarized in Table III. The model training and testing time generally increases with length of input and target sequences for equal amount of dataset. Given that amount of our training dataset reduces for longer sequences, the results show decrease in training time with increase in q. This variation in dataset is because the sequences of required lengths are created from same amount of raw handover logs, and the increase in target sequence length decreases the number of sequences in dataset. The time for model testing is significantly less than training, since the testing data is smaller than the training data, and backpropagation process is removed at the testing phase. In particular, testing time of ED is higher than SFI due to $3q \cdot r$ computation time of decoder network and its implementation in Keras library. The decoder of GRU-ATTN has similar computation processes to ED with an additional attention mechanism, thus it take more testing time than ED. However, the training time of GRU-ATTN model is insignificantly less than ED model, since GRU learning unit has reduced gate calculations than LSTM. Comparing to other models, TN takes considerable training time due to six layers of attention and fully-connected layers. The network requires mobile path prediction for only few mobile users at a time, and to this end Table III presents testing time for single sequence (*i.e.*, inference time). It shows that SFI predicts the target sequence within 0.7 to 1.1ms which satisfies latency requirements of 5G mobility management, while ED takes

TABLE III
TRAINING AND TESTING TIME COMPARISON OF FOUR MODELS FOR VARIOUS TARGET SEQUENCE LENGTHS WITH INPUT SEQUENCE LENGTH 22

Target Sequence Length q	Number of Sequences for Training, Testing	Training Time (sec)				Testing Time (sec)				Testing Time per Sequence (sec)			
		SFI	ED	GRU-ATTN	TN	SFI	ED	GRU-ATTN	TN	SFI	ED	GRU-ATTN	TN
3	2179, 933	151.85	67.63	61.59	353.8	0.64	137.23	300.95	19.76	0.0007	0.1471	0.3226	0.0212
5	1973, 845	133.43	64.54	60.837	346.3	0.72	177.27	407.16	25.00	0.0009	0.2098	0.4818	0.0296
7	1803, 772	126.70	62.25	58.60	336.5	0.79	210.85	501.04	30.37	0.0011	0.2731	0.6790	0.0393

147 to 273 *ms* for same operations. The inference time of GRU-ATTN is notably increased due to attention calculation, while TN has the lower inference time through the parallelized attention computations (*e.g.*, multi-head attention). Based on these results, average max unit computation time r for SFI and ED can be quantified as 7.9µ*s* and 5.28*ms*, respectively. In our finding, this difference in r of two models is down to implementation differences and use of GPU for computation in Keras library.

V. CONCLUSION AND FUTURE WORK

This paper proposes SFI and ED models based on recursive and multi-output approaches, respectively, for time-series mobile path prediction that enables proactive mobility and strategic MEC management in 5G. Training and testing of the models is conducted on preprocessed sequences of user movement which are collected from wireless network of Pangyo reach center in the form of handover logs. The experiments are done for the target path sequences of lengths 3, 5, and 7, as this information is sufficient for preemptive mobility and MEC resource management. Overall accuracy result and in-depth step wise analysis show that high accuracy of single-step prediction model in SFI model successfully curtails the accumulated errors till 5^{th} step of target sequence and has better results than ED model. The capability of ED model to capture temporal dependencies for predicting simultaneous multiple outputs becomes prominent for 6^{th} and 7^{th} step of target sequence as it shows higher accuracy than SFI model. In terms of consistency and reliability of both models, precision analysis of results exhibits that SFI model performs more consistently than ED model. Furthermore, the ED model requires shorter training time than the SFI model, but employs longer time for model inference. This concludes that SFI model is more suited for proactive mobility where shorter target sequence is sufficient, while the ED model is more appropriate for MEC resource management due to its higher accuracy for longer target sequences despite of its limitations in terms of consistency and time complexity.

We are currently extending both models with self-attention mechanisms to exploit spatial and temporal domains to not only predict mobility patterns but also the estimated dwell time of users in the predicted PoAs. The model extensions are designed using multi-branch architecture with joint loss function for PoA and time prediction. Moreover, the proposed SFI and ED are used in our Deep Reinforcement Learning (DRL) based service migration framework that is currently under implementation, and in our work on dynamic tracking area for users in mobile networks.

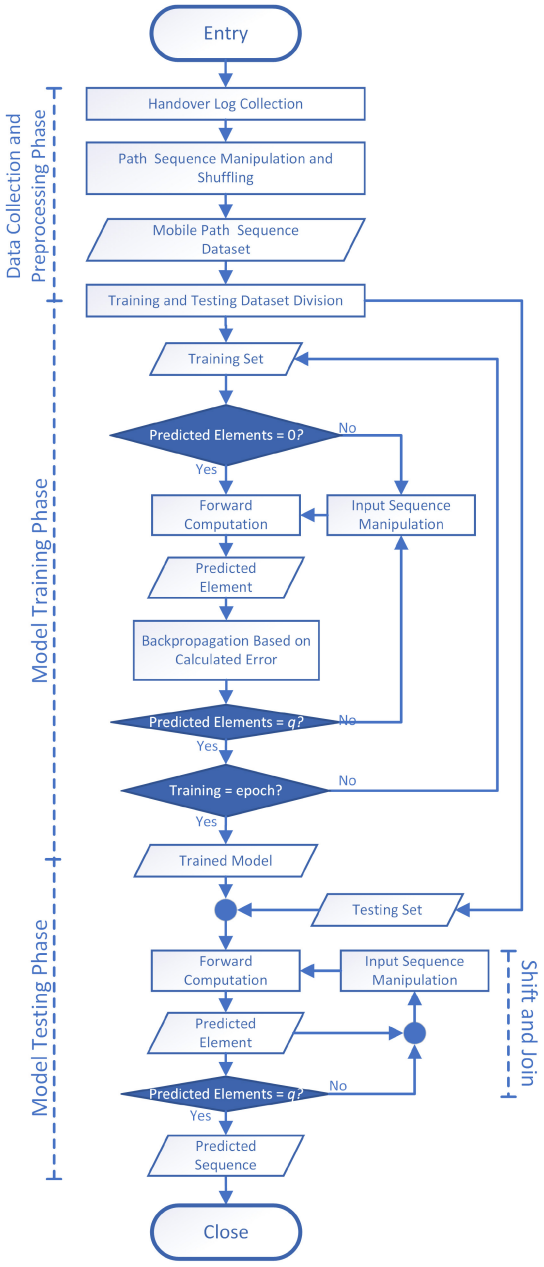


Fig. 11. The operational flowchart of the SFI model.

APPENDIX

The operational flowcharts of both SFI and ED models are shown in Fig. 11 and Fig. 12 respectively, that explains sequence of different operations along with various conditions and loops in training and the testing phases

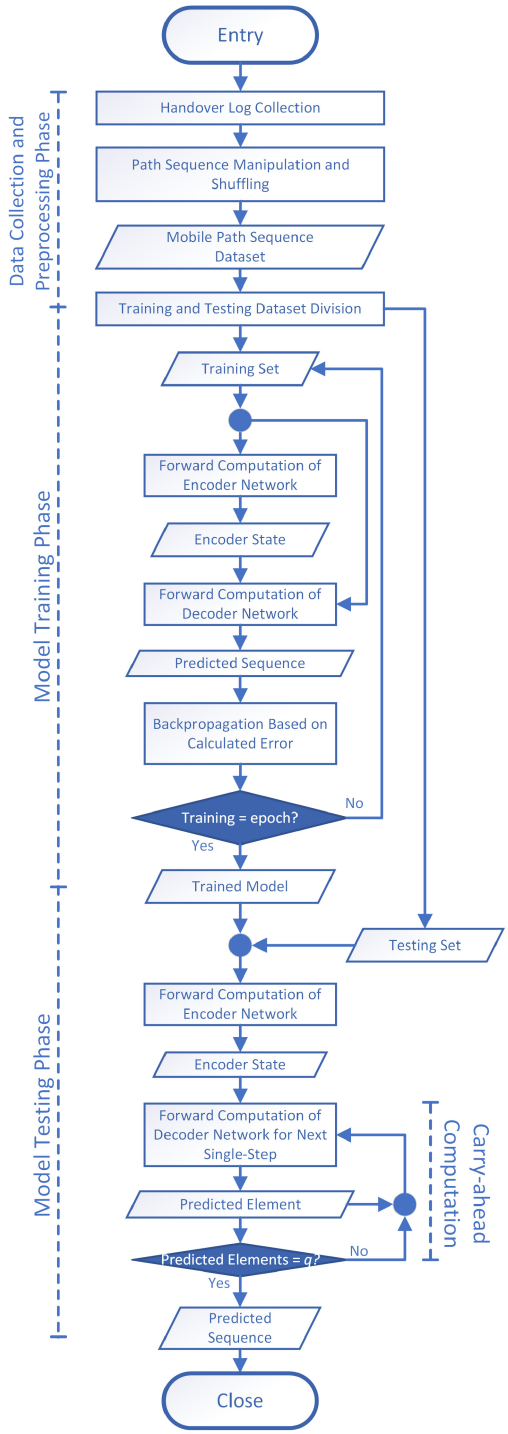


Fig. 12. The operational flowchart of the ED model.

REFERENCES

[1] J. Li, X. Zhang, J. Zhang, J. Wu, Q. Sun, and Y. Xie, "Deep reinforcement learning-based mobility-aware robust proactive resource allocation in heterogeneous networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 1, pp. 408–421, Mar. 2020.

[2] Y. Xun, J. Liu, and Z. Shi, "Multitask learning assisted driver identity authentication and driving behavior evaluation," *IEEE Trans. Ind. Informat.*, vol. 17, no. 10, pp. 7093–7102, Oct. 2021.

[3] Y. Xun, J. Qin, and J. Liu, "Deep learning enhanced driving behavior evaluation based on vehicle-edge-cloud architecture," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 6172–6177, Jun. 2021.

[4] Y. Qiao, Z. Si, Y. Zhang, F. B. Abdesslem, X. Zhang, and J. Yang, "A hybrid Markov-based model for human mobility prediction," *Neurocomputing*, vol. 278, pp. 99–109, Feb. 2018.

[5] Y. Ye, M. Xiao, and M. Skoglund, "Mobility-aware content preference learning in decentralized caching networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 1, pp. 62–73, Mar. 2020.

[6] 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 Trans. Veh. Technol.*, vol. 66, no. 6, pp. 5204–5216, Jun. 2017.

[7] 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, Sep. 2019.

[8] H. Yang, S. M. Raza, M. Kim, D.-T. Le, V. Van Vo, and H. Choo, "Next point-of-attachment selection based on long short term memory model in wireless networks," in *Proc. 14th Int. Conf. Ubiquitous Inf. Manage. Commun. (IMCOM)*, 2020, pp. 1–4.

[9] X. Ge, J. Ye, Y. Yang, and Q. Li, "User mobility evaluation for 5G small cell networks based on individual mobility model," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 3, pp. 528–541, Mar. 2016.

[10] R. He, B. Ai, G. L. Stüber, and Z. Zhong, "Mobility model-based non-stationary mobile-to-mobile channel modeling," *IEEE Trans. Wireless Commun.*, vol. 17, no. 7, pp. 4388–4400, Jul. 2018.

[11] K. Lee, S. Hong, S. J. Kim, I. Rhee, and S. Chong, "SLAW: Self-similar least-action human walk," *IEEE/ACM Trans. Netw.*, vol. 20, no. 2, pp. 515–529, Apr. 2012.

[12] 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, 2018.

[13] "The raw data for handover logs." Accessed: Aug. 4, 2021. [Online]. Available: http://monet.skku.edu/?page_id=13744

[14] M. Marcellino, J. H. Stock, and M. W. Watson, "A comparison of direct and iterated multistep AR methods for forecasting macroeconomic time series," *J. Econometr.*, vol. 135, nos. 1–2, pp. 499–526, 2006.

[15] C. Wang, L. Ma, R. Li, T. S. Durrani, and H. Zhang, "Exploring trajectory prediction through machine learning methods," *IEEE Access*, vol. 7, pp. 101441–101452, 2019.

[16] A. Koesdwiady, A. E. Khatib, and F. Karray, "Methods to improve multi-step time series prediction," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2018, pp. 1–8.

[17] G. Chevillon and D. F. Hendry, "Non-parametric direct multi-step estimation for forecasting economic processes," *Int. J. Forecast.*, vol. 21, no. 2, pp. 201–218, 2005.

[18] G. Chevillon, "Direct multi-step estimation and forecasting," *J. Econ. Surveys*, vol. 21, no. 4, pp. 746–785, 2007.

[19] M. W. McCracken and J. T. McGillicuddy, "An empirical investigation of direct and iterated multistep conditional forecasts," *J. Appl. Econometr.*, vol. 34, no. 2, pp. 181–204, 2019.

[20] G. Xue, Z. Li, H. Zhu, and Y. Liu, "Traffic-known urban vehicular route prediction based on partial mobility patterns," in *Proc. 15th Int. Conf. Parallel Distrib. Syst.*, 2009, pp. 369–375.

[21] C. Hamzaçebi, D. Akay, and F. Kutay, "Comparison of direct and iterative artificial neural network forecast approaches in multi-periodic time series forecasting," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 3839–3844, 2009.

[22] S. B. Taieb and A. F. Atiya, "A bias and variance analysis for multistep-ahead time series forecasting," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 1, pp. 62–76, Jan. 2016.

[23] A. Venkatraman, M. Hebert, and J. A. Bagnell, "Improving multi-step prediction of learned time series models," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 1–7.

[24] N. H. An and D. T. Anh, "Comparison of strategies for multi-step-ahead prediction of time series using neural network," in *Proc. Int. Conf. Adv. Comput. Appl. (ACOMP)*, 2015, pp. 142–149.

[25] H. Si, Y. Wang, J. Yuan, and X. Shan, "Mobility prediction in cellular network using hidden Markov model," in *Proc. 7th IEEE Consum. Commun. Netw. Conf.*, 2010, pp. 1–5.

[26] Y. Yang, W. Liu, E. Wang, and J. Wu, "A prediction-based user selection framework for heterogeneous mobile crowdSensing," *IEEE Trans. Mobile Comput.*, vol. 18, no. 11, pp. 2460–2473, Nov. 2019.

[27] J. Liu and E. Zio, "SVM hyperparameters tuning for recursive multi-step-ahead prediction," *Neural Comput. Appl.*, vol. 28, no. 12, pp. 3749–3763, 2017.

[28] A. Fandango and R. P. Wiegand, "Towards investigation of iterative strategy for data mining of short-term traffic flow with recurrent neural networks," in *Proc. 2nd Int. Conf. Inf. Syst. Data Min.*, 2018, pp. 65–69.

[29] S. M. Miran, S. J. Nelson, D. Redd, and Q. Zeng-Treitler, "Using multivariate long short-term memory neural network to detect aberrant signals in health data for quality assurance," *Int. J. Med. Inform.*, vol. 147, Mar. 2021, Art. no. 104368.
[30] X. Wang and Y. Zhang, "Multi-step-ahead time series prediction method with stacking LSTM neural network," in *Proc. 3rd Int. Conf. Artif. Intell. Big Data (ICAIBD)*, 2020, pp. 51–55.
[31] S. Hexeberg, A. L. Flåten, B.-O. H. Eriksen, and E. F. Brekke, "AIS-based vessel trajectory prediction," in *Proc. 20th Int. Conf. Inf. Fusion (Fusion)*, 2017, pp. 1–8.
[32] Y. Zheng, X. Xie, and W.-Y. Ma, "GeoLife: A collaborative social networking service among user, location and trajectory," *IEEE Data Eng. Bull.*, vol. 33, no. 2, pp. 32–39, Jun. 2010.
[33] M. H. Wong, V. S. Tseng, J. C. Tseng, S.-W. Liu, and C.-H. Tsai, "Long-term user location prediction using deep learning and periodic pattern mining," in *Proc. Int. Conf. Adv. Data Min. Appl.*, 2017, pp. 582–594.
[34] M. Lv, D. Zeng, L. Chen, T. Chen, T. Zhu, and S. Ji, "Private cell-ID trajectory prediction using multi-graph embedding and encoder-decoder network," *IEEE Trans. Mobile Comput.*, vol. 21, no. 8, pp. 2967–2977, Aug. 2022.
[35] J. Zheng, C. Xu, Z. Zhang, and X. Li, "Electric load forecasting in smart grids using long-short-term-memory based recurrent neural network," in *Proc. 51st Annu. Conf. Inf. Sci. Syst. (CISS)*, 2017, pp. 1–6.
[36] Y. Zhou, F.-J. Chang, L.-C. Chang, I.-F. Kao, Y.-S. Wang, and C.-C. Kang, "Multi-output support vector machine for regional multi-step-ahead $PM_{2.5}$ forecasting," *Sci. Total Environ.*, vol. 651, pp. 230–240, Feb. 2019.
[37] H. Cheng, Z. Xie, Y. Shi, and N. Xiong, "Multi-step data prediction in wireless sensor networks based on one-dimensional CNN and bidirectional LSTM," *IEEE Access*, vol. 7, pp. 117883–117896, 2019.
[38] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
[39] G. Neubig, "Neural machine translation and sequence-to-sequence models: A tutorial," 2017, *arXiv:1703.01619*.
[40] L. Ma and S. Qu, "A sequence to sequence learning based car-following model for multi-step predictions considering reaction delay," *Transp. Res. C, Emerg. Technol.*, vol. 120, Nov. 2020, Art. no. 102785.
[41] S. Wang, J. Cao, H. Chen, H. Peng, and Z. Huang, "SeqST-GAN: Seq2Seq generative adversarial nets for multi-step urban crowd flow prediction," *ACM Trans. Spatial Algorithms Syst.*, vol. 6, no. 4, pp. 1–24, 2020.
[42] J. J.-C. Ying, W.-C. Lee, T.-C. Weng, and V. S. Tseng, "Semantic trajectory mining for location prediction," in *Proc. 19th ACM SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst.*, 2011, pp. 34–43.
[43] J. Feng et al., "DeepMove: Predicting human mobility with attentional recurrent networks," in *Proc. World Wide Web Conf.*, 2018, pp. 1459–1468.
[44] Y. Filippas, A. Margaris, and K. Tsagkaris, "Deep learning approaches for mobile trajectory prediction," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, 2021, pp. 1–6.
[45] E. Chaalal, S.-M. Senouci, and L. Reynaud, "A new framework for multi-hop ABS-assisted 5G-networks with usersx2019; mobility prediction," *IEEE Trans. Veh. Technol.*, vol. 71, no. 4, pp. 4412–4427, Apr. 2022.
[46] R. J. Williams and D. Zipser, "Gradient-based learning algorithms for recurrent," in *Backpropagation: Theory, Architectures, and Applications*. Mahwah, NJ, USA: Lawrence Erlbaum Assoc., 1995, p. 433–486.
[47] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: The difficulty of learning long-term dependencies," in *A Field Guide to Dynamical Recurrent Neural Networks*. New York, NY, USA: IEEE Press, 2001.
[48] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[49] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.
[50] "Pangyo ICT research center." Accessed: Jun. 28, 2021. [Online]. Available: http://gitrc.kr
[51] L. Zhang, H. Zhang, Y. Jiang, and Z. Wu, "Intelligent and reliable deep learning LSTM neural networks-based OFDM-DCSK demodulation design," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16163–16167, Dec. 2020.
[52] G. de Veciana and A. Zakhor, "Neural net-based continuous phase modulation receivers," *IEEE Trans. Commun.*, vol. 40, no. 8, pp. 1396–1408, Aug. 1992.
[53] S. Pulipaka and R. Kumar, "Comparison of SOM and conventional neural network data division for PV reliability power prediction," in *Proc. IEEE Int. Conf. Environ. Electr. Eng. IEEE Ind. Commercial Power Syst. Europe (EEEIC/ICPS Europe)*, 2017, pp. 1–5.

Huigyu Yang received the B.S. degree in computer engineering from Chungbuk National University, South Korea, in February 2019, and the M.S. degree in electrical and computer engineering from Sungkyunkwan University, South Korea, in February 2021, where he is currently pursuing the Ph.D. degree with the Department of Super-Intelligence. His current research interests include proactive mobility, network data manipulation, and AI driven intelligent networking in nextgen wireless networks.

Syed M. Raza received the master's degree from Lund University, Sweden, in 2009, and the Doctoral degree from Sungkyunkwan University, South Korea, in 2018, where he has been a Research Professor with the Department of Electrical and Computer Engineering since 2019. Before that, he was a Postdoctoral Fellow with the College of Software, Sungkyunkwan University from 2018 to 2019. Prior to his Doctoral degree, he had been a Lecturer with the department of Computer Science, Comsats University, Pakistan, from 2011 to 2012. His current research interests include intelligent networks, network softwarization, and in network computing.

Moonseong Kim received the M.S. degree in mathematics and the Ph.D. degree in electrical and computer engineering from Sungkyunkwan University, South Korea, in August 2002 and February 2007, respectively, where he was a Research Professor in 2007. From December 2007 to October 2009, he was a Research Associate with the Department of ECE and the Department of CSE, Michigan State University, USA. He was a Deputy Director and a Patent Examiner with Korean Intellectual Property Office, Daejeon, South Korea, from October 2009 to August 2018. In September 2018, he joined Seoul Theological University, Bucheon, South Korea, where he is currently working as an Assistant Professor and the Head of the Department of IT Convergence Software. His research interests include wired/wireless networking, sensor networking, mobile computing, network security protocols, and simulations/numerical analysis.

Hyunseung Choo (Member, IEEE) received the B.S. degree from Sungkyunkwan University (SKKU), South Korea, in 1988, the M.S. degree from The University of Texas at Dallas, USA, in 1990, and the Ph.D. degree from The University of Texas at Arlington, USA, in 1996. He is a Professor with College of Computing and Informatics, SKKU, and Director of the ICT Creative Consilience Program supported by the Ministry of Science and ICT (MIST), South Korea. Previously, he was the Director of the Intelligent HCI Convergence Research Center supported by the Ministry of Knowledge Economy, South Korea. He has also served as a Technical Adviser for the Samsung Electronics DMC Research and Development Center (next generation interaction). He specializes in network softwarization, intelligent mobile computing, and multi-access edge computing, with over 350 publications in international journals and refereed conferences, and 21 international (USA) and 208 domestic patents in South Korea, in the field of mobile and sensor networks with intelligence and autonomicity. For his outstanding research, he has received two excellence awards and one commendation award over the years by MIST. He has been the Editor-in-Chief of the Journal of Korean Society for Internet Information for three years, a Journal Editor of *ACM Transactions on Internet Technology*, *Journal of Communications and Networks*, and *Journal of Supercomputing*, and the Founding Editor of Transactions on Internet and Information Systems since 2010. He is a member of the ACM and IEICE.