Location Embedding and Deep Convolutional Neural Networks for Next Location Prediction

Abdessamed Sassi, Mohammed Brahimi, Walid Bechkit, Abdelmalik Bachir

Location Embedding and Deep Convolutional Neural Networks for Next Location Prediction

Abdessamed Sassi*†‡ Mohammed Brahimi§¶, Walid Bechkit*, Abdelmalik Bachir‡
* Univ Lyon, Inria, INSA Lyon, CITI, F-69621 Villeurbanne, France
† Computer science department, Oum El Bouaghi University, Algeria.
‡ Computer science department, Biskra University, Algeria.
§ Computer science department, Bordj Bou Arreridj University, Algeria.
¶ Computer science department, USTHB University, Algeria.

***Abstract*—We focus in this work on predicting the next location of mobile users by analyzing large data sets of the history of their movements. We make use of past location sequences to train a classification model that will be used to predict future locations. Contrary to traditional mobility prediction techniques based on Markovian models, we investigate the use of modern deep learning techniques such as the use of Convolutional Neural Networks (CNNs). Inspired by the word2vec embedding technique used for the next word prediction, we present a new method called loc2vec in which each location is encoded as a vector whereby the more often two locations cooccur in the location sequences, the closer their vectors will be. Using the vector representation, we divide long mobility sequences into several sub-sequences and use them to form Mobility Subsequence Matrices on which we run CNN classification which will be used later for the prediction. We run extensive testing and experimentation on a subset of a large real mobility trace database made publicly available through the CRAWDAD project. Our results show that loc2vec embedding and CNN-based prediction provide significant improvement in the next location prediction accuracy compared to state-of-the-art methods. We also show that transfer learning on existing pre-trained CNN models provides further improvement over CNN models build from scratch on mobility data. We also show that our loc2vec-CNN model enhanced with transfer learning achieves better results than other variants including our other proposal onehot-CNN and existing Markovian models.**

***Index Terms*—Next Location Prediction, Location Embedding, Convolutional Neural Networks, WiFi Mobility Traces.**

I. Introduction

The ability to accurately predict the future locations of mobile users has become a fundamental requirement for a wide range of location-based services in many areas including urban management, travel recommendation systems [1]–[3], advertisement dissemination [4], leisure event reporting [5], intelligent HVAC systems [6], etc.

Predicting future locations of a user is generally based on analyzing the history of their locations and identifying repeating patterns [7]. Many techniques have been used to achieve this. Markovian models are one of the models that have been extensively used in the literature [8]–[12]. Markovian models assume that the next location of a user depends on the current context defined as a sequence of the most recent locations visited by the user. According to the considered context length, various accuracy values have been obtained for different situations. Finding the optimal context length that achieves high accuracy values consistently has been one of the main limitations the Markovian models.

With the recent advances in machine learning, new next location prediction algorithms have been developed based on sequence modeling with deep neural networks [13]–[15]. In these models, the locations visited by a user are considered as a sequence of elements. Prediction of the next location is thus seen as a pattern recognition problem which can be solved using a neural network.

Neural Networks have been generally used efficiently in sequence modeling and next element prediction in many application domains. For example, in [14], the authors developed a neural networks model for the prediction of the next word in a text. In their approach [16], they enhanced the performance of their neural network model by making use of a new embedding technique which consists in encoding words as vectors of real values. The proposed embedding technique called word2vec allowed to achieve significant performance improvement as it ensures that the distance between the vectors representing words reflects the closeness of these words in text documents, i.e. words that tend to be next to each other frequently have very close vector representations.

In our paper, we follow a similar approach as word2vec and propose a new location embedding technique that we use on locations instead of words. Similar to word2vec philosophy, loc2vec embedding ensures that locations that are likely to appear close to each other in location sequences (i.e. locations frequently seen the one next to the other) are embedded into vectors such that the distance between these vectors is small.

In our solution, we rely on Convolutional Neural Networks (CNNs) and propose new prediction techniques called onehot-CNN and loc2vec-CNN which are built by converting mobility sequences into matrices which we call *Mobility Sequence Matrices*. These matrices could be seen as similar to images and thus allow us to make use of CNN classifiers particularly those which have been pre-trained on ImageNet datasets. The use of pre-trained CNN models has achieved significant results in various application domains such as sequence modeling [17], malware binary detection [18], action recognition [19], [20], sounds classification [21], etc. compared to CNN models established from scratch on the specific data of the considered domain.

The main contributions of this paper are the following:

- Traditional next location prediction algorithms are based on a symbolic representation of locations in a way they consider each location as a different symbol. With such a representation, it is not easy to include more information that provides additional meaningful and helpful description for the location. The proposed loc2vec location embedding technique, however, represents each location as a vector by taking into consideration several features. In our case we consider, to represent locations as vectors, the surrounding locations, i.e. previous and next location. A better prediction results can be achieved by integrating loc2vec in the prediction model.
- Traditional prediction models such as those based on Markov chains do not perform well with long sequences, and cannot build a robust prediction model that is not highly dependent on context length. We propose an innovative representation of mobility subsequences which we call Mobility Subsequence Matrix which allows having a two-dimensional representation of mobility subsequences and thus can be used as inputs for a CNN model that has been pre-trained on large image datasets such as ImageNet [22]. With the Mobility Subsequence Matrices representation, we propose two variants of CNN-based location prediction algorithms called onehot-CNN and loc2vec-CNN which are based on onehot and loc2vec location representations respectively.
- We conduct extensive simulations on large real mobility traces from the CRAWDAD project [23], [24] and compare our results with those obtained by means of the prediction techniques based on Markovian models. Experimentation results show that we achieve a stable and higher prediction accuracy compared to those proposed in the literature.

The remainder of the paper is organized as follows. In Section II, we provide an overview of the main contributions on solving the next location prediction problem. In Section III, we present our prediction model and algorithms. In Section IV, we evaluate the performance of our algorithms and discuss the obtained results. In Section V, we conclude the paper by summarizing our findings and discussing options for future work.

II. Related Work

A variety of algorithms for next location prediction have been proposed in the literature. Most algorithms focused on Markovian models (e.g. [9], [11]). These models, called Order-k ($O(k)$) Markovian models, assumed that the probability of visiting a particular next location depends on the current context defined as the sequence of the k most recent locations visited.

In [8], the authors evaluated and compared the prediction accuracy of several $O(k)$ Markovian location predictors with the goal of enhancing the initial Markovian model with a simple fallback mechanism by decrementing the order of the model to $k-1$, then $k-2$, ..., in case the searched context has not been seen before in the location history. In their experiments, they found that lower order Markov predictors provide more accurate results compared to higher order ones. In particular, the $O(2)$ Markovian model with fallback has been shown to be the best overall predictor.

In [11], the authors proposed a next location prediction algorithm called n-MMC, by using the Mobility Markov Chains (MMC) presented in [12]. This model incorporates the n previously visited locations to predict the next location of users. The evaluation over different datasets showed that a high accuracy for the next location prediction is obtained with $n = 2$. Similar to the results obtained with Order-k Markovian models, the authors showed that having a context larger than 2 elements did not improve the accuracy of the prediction.

The previously mentioned models have many advantages. They are easy to implement and do not require large memory space as the predictor updates only one transition probability after each movement from one location to another. These models have however some limitations caused by the difficulty of a priori finding the best value for k as these vary from a situation to another. With a large value for k in a Markovian model, chances of encountering a pattern that has been seen before in the location history are slim and thus a prediction cannot be done without performing a fallback into lower values for k.

In [25], the authors presented a prediction model in which they ran several parallel predictors and performed voting to select the best predictors. In their proposal they incorporated more information to the Markovian model (the time of the day, the day of the week, etc.). They derived several prediction algorithms by using different combinations of spatial and temporal features. The authors proposed an algorithm, called Major, which predicts the next location of the user based on a voting processes combining the outputs of the several prediction algorithms used.

In [15], the authors treated the problem of next location prediction as a classification problem. They proposed a solution based Long Short Term Memory (LSTM) neural networks which is a class of Reccurent Neural Networks (RNNs). Our proposal differs from the aforementioned work on two aspects. First, we consider a CNN architecture as opposed to RNN, which allows us to take advantage of transfer learning from pre-trained CNNs. Second, our work makes use of location embedding according to the loc2vec technique which contributes in improving the quality of learning and prediction.

III. Proposed Approach

We assume in this work that, at any given time, a user resides at a given discrete location. We assume $\mathcal{L}$ is the set of all discrete locations where $\mathcal{L} = \{l_1, l_2, \ldots, l_i, \ldots, l_n\}$. In our data, the location is expressed as the access point with which the user device is associated (i.e., there are n different access points). Our aim is to answer the question: where will a user go next. First, we describe the proposed approach, illustrated in Figure 1, which contains three components as follows:

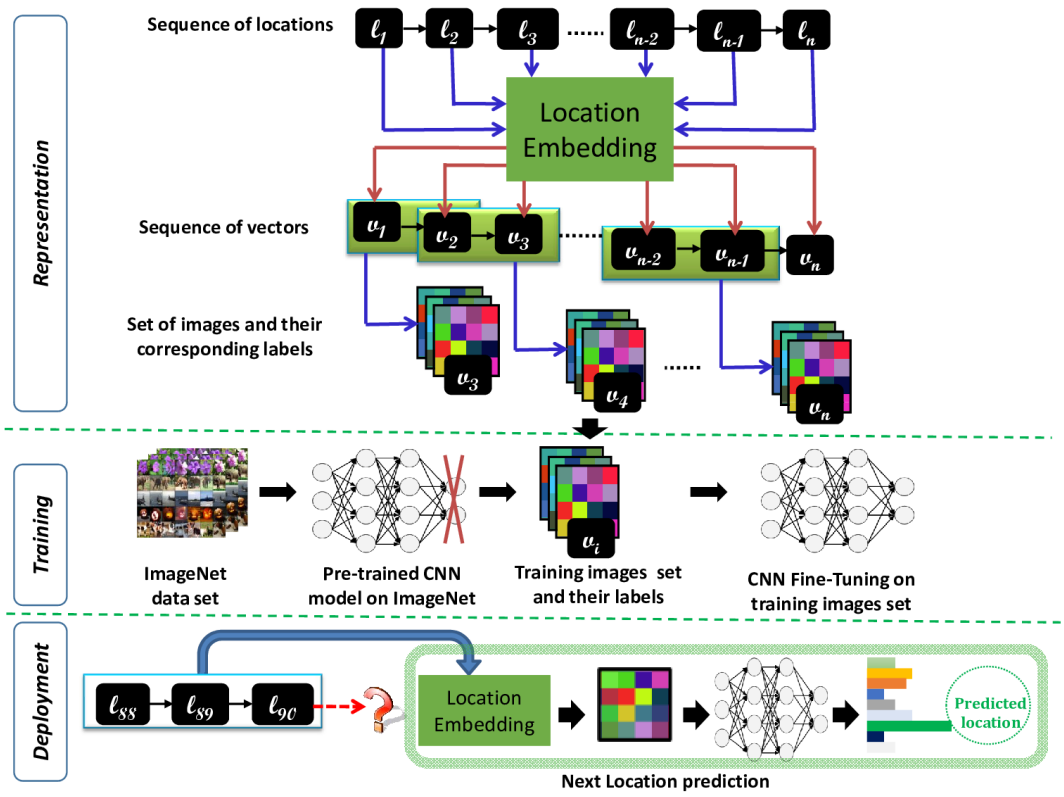

Fig. 1: Overview of next location prediction based on CNN.

1) Representation phase: In this phase, we perform location embedding which consists in representing each location $l_i \in \mathcal{L}$ by a vector $\mathbf{v_i}$ of length l where $\mathbf{v_i}$ is defined as $\mathbf{v_i} = (v_{i1}, v_{i2}, \ldots, v_{il})^T$. For a sequence of successive user locations, we consider sequence embedding by replacing each location in the sequence by its corresponding vector. After that, we divide the embedded location sequence (i.e. a sequence of vectors $\mathbf{v_i}$) into multiple sub-sequences with fixed length k. Finally, each subsequence can be represented as a matrix and thus can be seen as an image. We give a label for each subsequence (or image), the label is the next vector in the location embedding sequence. For example: consider a location sequence of a given user as $l_1 l_2 l_3 l_4 l_5$, the corresponding embedding will be $\mathbf{v_1 v_2 v_3 v_4 v_5}$. If we take a window length of $k = 2$ to construct sub-sequences, we get the following sub-sequences (in the case where we do not take overlapping sub-sequences) $\mathbf{v_1 v_2}$ and $\mathbf{v_3 v_4}$. We provide a label for each subsequence in the following way: the label of a given sub-sequence is the embedding vector representing the next location in the original sequence, i.e. for the sub-sequence $\mathbf{v_1 v_2}$ the label will be $\mathbf{v_3}$ and for the other subsequence $\mathbf{v_3 v_4}$ the label will be $\mathbf{v_5}$. Note that sub-sequencing can also generate overlapping sub-sequences as shown in Table II.
The output of this phase is a set of images classified according to their next location labels. We divide the set of images assigned to each label into two parts. We use the first part to train the model and the second part to test it.
2) Training phase: we propose to use a CNN to train an image classification model that will be used to predict future locations.

- Pre-training: rather than starting from a model with a random configuration of neural network parameters, we can instead start with parameters of an already trained neural network on a specific task with a very large number of examples [18], [19], [21], [26]. In this phase, deep architectures will be trained on a large dataset of images like ImageNet [22] using powerful machines with the aim of initializing the network weights to be used for the next phase. This pre-training phase is optional and we evaluate later in this work the improvement while using pre-training. The pre-training phase results in a neural network composed of an input layer, middle layers, and an output layer with the weights set for the connection between the elements of the neural network. The output layer represents

the number of classes in the classification problem to be solved.
- Training (fine-tuning): by changing the number of classes of the output layer of the pre-trained model to match the number of next locations, we fine-tune the model by considering a part of images set, i.e the training images set.
- Testing: we test our model by considering the second part of images set.

3) Deployment: the obtained CNN can be used to predict the next location of users by finding the corresponding label (which represents the next location) from a given sub-sequence of user history locations.

A. Embedding Methods for Location Representations

Embedding methods, which correspond to representing a given piece of data by a vector of reals, are being extensively used as inputs to machine learning algorithms, especially in the deep learning community [16], [27], [28]. Several embedding methods have been proposed in the literature [29] with one-hot embedding and word embedding being the most popular ones. We consider both methods of embedding with our proposal based on pre-trained CNNs which we name one-hot and loc2vec.

1) One-Hot Representation: In the one-hot embedding, a given location $l_i \in \mathcal{L}$ is represented by a vector $\mathbf{v_i^{onehot}}$ which the dimension l is equal to n where l_1 is embedded as $\mathbf{v_1^{onehot}}$ with $\mathbf{v_1^{onehot}} = (1, 0, \ldots, 0)^T$, l_2 is embedded as $\mathbf{v_2^{onehot}}$ with $\mathbf{v_2^{onehot}} = (0, 1, \ldots, 0)^T$, and l_n is embedded as $\mathbf{v_n^{onehot}}$ with $\mathbf{v_n^{onehot}} = (0, 0, \ldots, 1)^T$. In general, a location l_i is represented by a vector $\mathbf{v_i^{onehot}} = (v_1, v_2, \ldots, v_j, \ldots, v_n)^T$ where:

$$v_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \quad j \in \{1, \ldots, n\}$$

2) loc2vec Representation: The previously described representation one-Hot does not take context into consideration. Therefore, we propose another representation inspired from word2vec [16] to take context into account which we call loc2vec. In order to find the best embedding, loc2vec uses a similar approach as word2vec which consists in using a neural network with a single hidden layer as shown in Figure 2 to find the best vector corresponding to a given location. The basic idea behind loc2vec is to attribute to each location l_i a vector $\mathbf{v_i^{loc2vec}}$ such as locations that have the same neighboring locations in sequences will have similar embedding. For example, given a hypothetical sequence of locations $l_1 l_2 l_3 l_4 l_1 l_5 l_3$, the locations l_2 and l_5 are surrounded by the same locations l_1 and l_3. Therefore, the locations l_2 and l_5 will have similar vectors $\mathbf{v_2^{loc2vec}}$ and $\mathbf{v_5^{loc2vec}}$ respectively.

We propose to build our loc2vec neural network on locations and their surrounding locations (i.e previous and next locations in the mobility sequence). For a given location l_i the surrounding locations are all location l_j defined such that $j \in \{i-c, \ldots, i-2, i-1, i+1, i+2, \ldots, i+c\}$ where c is the length of the context, which delimits how long the surrounding context is taken around the location i. The loc2vec neural network is built by training it on various pairs (l_i, l_j). We show in Table I some of the training examples (location pairs) taken from location sub-sequence of length equal to 9 and a context c of length equal to 2.

TABLE I: Training examples extracted from a sub-sequence with the loc2vec neural network model

Sub-sequence of locations	Training examples (l_i, l_j)
$\mathbf{l_1}$ $\mathbf{l_2}$ $\mathbf{l_3}$ l_4 l_5 l_6 l_7 l_8 l_9	(l_1, l_2), (l_1, l_3)
$\mathbf{l_1}$ $\mathbf{l_2}$ $\mathbf{l_3}$ $\mathbf{l_4}$ l_5 l_6 l_7 l_8 l_9	(l_2, l_1), (l_2, l_3), (l_2, l_4)
$\mathbf{l_1}$ $\mathbf{l_2}$ $\mathbf{l_3}$ $\mathbf{l_4}$ $\mathbf{l_5}$ l_6 l_7 l_8 l_9	(l_3, l_1), (l_3, l_2), (l_3, l_4), (l_3, l_5)
...	...
l_1 l_2 l_3 l_4 l_5 l_6 $\mathbf{l_7}$ $\mathbf{l_8}$ $\mathbf{l_9}$	(l_9, l_7), (l_9, l_8)

The main goal of loc2vec is to represent a function that find the best association for each input location l_i with its corresponding output location l_j. To construct the hidden layer of the loc2vec neural network, we use all pairs (l_i, l_j). To achieve this, we consider the one-hot representations $\mathbf{v_i^{onehot}}$ $\mathbf{v_j^{onehot}}$ of locations l_i and l_j. In this paper, we propose two ways of generating sub-sequences from the locations sequence: non overlapping and overlapping sub-sequences. In the overlapping case, we generate an overlapping sub-sequences of fixed length s from the locations sequence by sliding a window of length s across the locations sequence. For example, given a sequence of locations $l_1 l_2 l_3 l_4 l_5 l_6 l_7 l_8$, a sub-sequence of length 5 with an overlapping of 2 locations can be generated as follows: $\{l_1 l_2 l_3 l_4 l_5, l_4 l_5 l_6 l_7 l_8\}$ In non-overlapping case, we generate sub-suequences of length s from the locations sequence while shifting the starting point by s consecutive locations at each step.

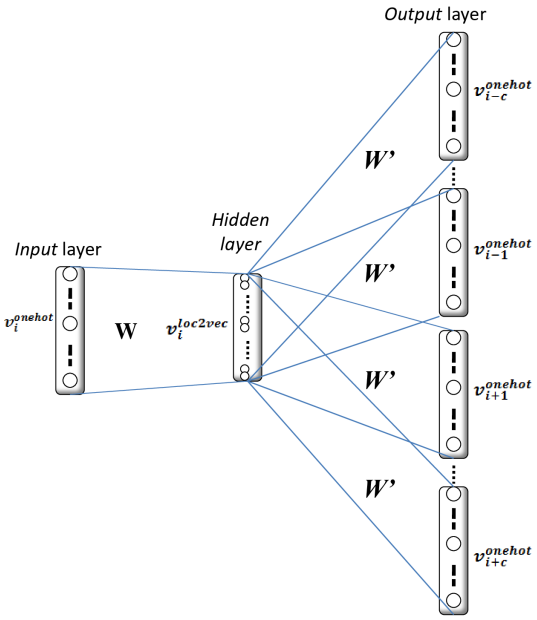

Fig. 2: Architecture of the training loc2vec neural network.

The goal of loc2vec is to find a representation vector $\mathbf{v_i^{loc2vec}}$ for each location l_i. By contrast to one-hot where the representation vector $\mathbf{v_i^{onehot}}$ of location l_i is of length n,

the length of the loc2vec representation vector $\mathbf{v_i^{loc2vec}}$ can be equal to a value m pre-set in advance and considered as a parameter which is the number of nodes in the hidden layer of the loc2vec neural network. To find the values of the vector $\mathbf{v_i^{loc2vec}}$, loc2vec consists in constructing a $n \times m$ matrix $\mathbf{W}$, called the weight matrix, where the i^{th} row of the matrix $\mathbf{W}$ represents the weights of location l_i. We have:

$$\mathbf{v_i^{loc2vec}} = \mathbf{W^T v_i^{onehot}} \quad (1)$$

where $\mathbf{W^T}$ is the transpose of matrix $\mathbf{W}$. The elements of the matrix $\mathbf{W}$ are constructed by applying back-propagation and stochastic gradient descent algorithms by considering the set of all inputs $l_i \in \mathcal{L}$ (represented by $\mathbf{v_i^{onehot}}$) and the corresponding outputs l_j (represented by $\mathbf{v_j^{onehot}}$) for each considered l_i.

B. Convolutional Neural Network for Next Location Prediction

Very good results are achieved using Convolutional Neural Networks (CNNs) in many areas in the literature [27], [30]–[32]. In this paper, we propose to use these powerful neural networks for next location prediction of users based on learning from previous mobility sequences.

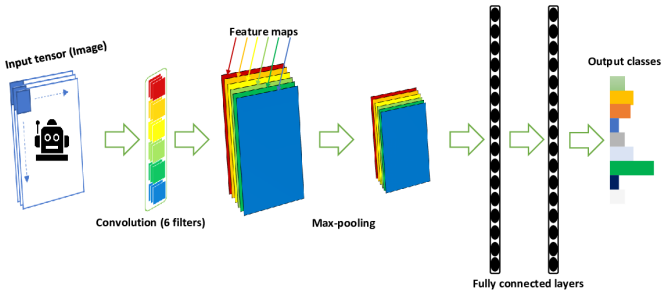

Fig. 3: Convolutional Neural Network (CNN)

The general architecture of a CNN is presented in Figure 3. A CNN is typically composed of a Convolution layer, Max-pooling layer and two Fully Connected layers. Depending on the way these layers are superposed, various architectures can be constructed providing different performance results for different application domains.

A CNN takes various types of inputs and provide results as outputs. Depending on the structure and the encoding of input, different results may be obtained.

As shown in Table II, we propose to encode inputs as Mobility Subsequence Matrices where each matrix is composed of a mobility subsequence of length k as explained in Section III. Each element of the mobility sub-sequence (i.e. each vector $\mathbf{v_i}$ of the Mobility Subsequence Matrix) is represented by one-hot embedding and loc2vec embedding resulting in two methods that we refer to as one-hot-CNN and loc2vec-CNN respectively.

Finally, an image and its label are generated from each Mobility Subsequence Matrix to be used by CNN. Both images and their labels (see Table II) are used to train a CNN model as well as making predictions. Hence, our model will take an image as an input and will output a label which correspond to the next location to be visited.

It is important to note that our loc2vec-CNN construction ensure a spatial locality in the image generation to get closer to the spatial structure of real images. Indeed, each row i of the image corresponds to the encoded vector of location l_i which is itself computed using a neural network based on the surrounding locations (see Section III-A). The vectors corresponding to rows after and before the row i are computed based on neighbouring surrounding locations which guarantee the spatial locality.

The success of using a given CNN solution depends on the architecture and also on the availability of pre-training. It has been shown that a CNN with pre-training generally provides significantly better results than a randomly initialized CNN. We consider using existing pre-trained CNN models that have been constructed based on a large image dataset like ImageNet (e.g. Inception [33], ResNet [34], SequeezeNet [35], DenseNet [36], etc.)) which showed significant results in various application domains [18]–[21], [26]. The rationale of relying on images is two-fold: (i) using a pre-trained network is generally better than a randomly initialized one even if the domain applications are different (e.g. environmental sound classification [18]), and (ii) there is a similarity between our Mobility Subsequence Matrices and images as there is proximity between neighboring pixels in images and neighboring location in a mobility subsequence.

In our solution, we choose to rely on SequeezeNet architecture [35] which is a CNN that has been trained on very large image datasets. In our solution, we use a pre-trained CNN model that we fine-tune by further training it on a subset of Mobility Subsequence Matrices (as inputs) and their corresponding labels (as outputs).

TABLE II: Generating Mobility Subsequence Matrices and their corresponding labels from a mobility sequence according to a sliding window of length $k = 3$

Sequence of vectors	Mobility Sub-sequence Matrix	Label
$\mathbf{v_1 v_2 v_3 \dots v_n}$	$(\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3})^T$	$\mathbf{v_4}$
$\mathbf{v_1 v_2 v_3 v_4 \dots v_{n-1} v_n}$	$(\mathbf{v_2}, \mathbf{v_3}, \mathbf{v_4})^T$	$\mathbf{v_5}$
$\dots$	$\dots$	$\dots$
$\mathbf{v_1 \dots v_{n-3} v_{n-2} v_{n-1} v_n}$	$(\mathbf{v_{n-3}}, \mathbf{v_{n-2}}, \mathbf{v_{n-1}})^T$	$\mathbf{v_n}$

IV. Performance Evaluation

A. Evaluation Dataset

The dataset used in this paper is a subset of the WLAN trace data extracted from Dartmouth College [37]. In this dataset, mobility sequence is expressed in the form of (time, location) pairs for each user where location is taken to be that of the access point to which the user is associated. In the dataset we are working on, there are more than 543 different access points resulting in more than 543 different locations. As users move around these locations, they generate different mobility sequences for different users which lengths vary widely from a user to another reaching several thousand movements for some users. We wrote simulation code in Python and used

Gensim [38] module to implement our location embedding approach loc2vec.

B. Effect of loc2vec Parameters Choice

We start by taking a subset of our dataset which we use for the generation of loc2vec embedding for locations. On this subset, we run loc2vec training algorithm by considering all pairs extracted from each mobility subsequence by considering both overlapping and non overlapping subsequences. We set the length of the context $c = 3$, i.e. for each location in the mobility sequence we consider its one and three-hop neighboring locations.

In the overlapping case, we choose to reduce the length of the initial mobility sequences taken from the considered subset by splitting long sequences into subsequences of equal length $s = 32$. However, for the non-overlapping case, we reduce the length of sequences by splitting long ones into subsequences of lengths covering mobility sequence of one day.

In Figure 4, we evaluate the effect of loc2vec construction on the performance of the final prediction algorithm that is based on CNN, called loc2vec-CNN. We use the accuracy metric defined as the ratio between the number of correct next location predictions and that of all next location predictions made. In the evaluation of the effect of loc2vec parameters choice, we consider the following parameters: (i) the embedding vector length m, and (ii) the length of the subsequences s obtained from the splitting of long sequences. We consider the following values (m, s) = (50, 32), (10, 32) and (10, 1 day), with the two first values concerning the overlapping case and the latter one concerning the non-overlapping one. We show that non-overlapping loc2vec-CNN with a variable subsequences length (s=1 day) provides the best performance over the other considered cases. The reason is that constructing subsequences according to a certain logic (activities during a day) captures better the relations between user movements and activities than taking fixed length subsequences.

We also show that having relatively shorter loc2vec vectors is likely to be more accurate than having longer ones as loc2vec-CNN (m=10, s=32) provides better results than a loc2vec-CNN (m=50, s=32).

We also show in Figure 4 that the accuracy of the combination of loc2vec with CNN is affected by the choice of parameter k which determines the length of the context that is taken into account for the prediction of the next location. We show that increasing the length of k decreases the accuracy of the prediction which is in concordance with early results on location prediction based on O(k) Markovian models.

C. Evaluating the Performance of CNN-based Predictions

We evaluate the performance of our CNN-based location prediction proposal, we choose two variants for location encoding: one-hot and loc2vec, thereby resulting in two different methods which we call onehot-CNN and loc2vec-CNN respectively. We proceed as the following. We construct a subset by randomly choosing different users whose mobility sequences vary widely in length with a minimum of 2500, a medium

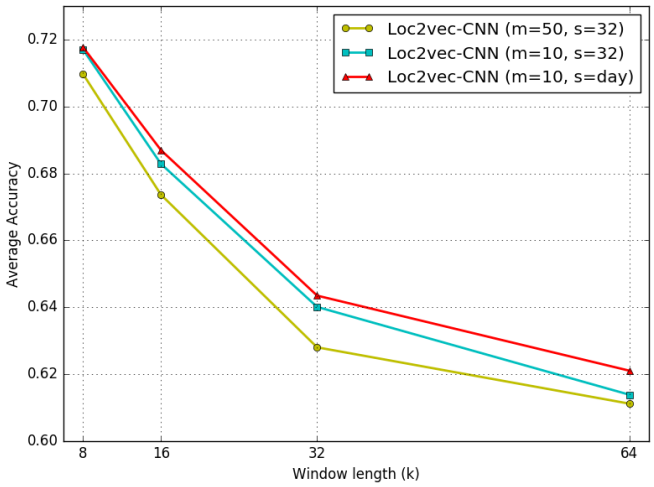

Fig. 4: Comparison of the three variants of loc2vec-CNN model according to the average accuracy metric with various subsequence lengths s and embedding vector sizes m.

of 6000, and a maximum of 13500 location sequences. For each user, we split the location sequence into two halves: we use the first half to build and train the model, and the second half as a live data to test the performance of our predictor. The training of the proposed CNN model is based on Stochastic gradient descent algorithm with the following hyperparameters: (learning rate = 0.001, momentum = 0.9, batch size= 20, number of epochs = 20).

We compare the performance of our models loc2vec-CNN and onehot-CNN with O(k) Markov predictor, which is one of the most popular next location predictors of the literature.

In Figure 5, we evaluate the performance of three models: loc2vec-CNN, onehot-CNN, and O(k) Markov, according to different context lengths by varying the value of k from 8 to 64. We show that the loc2vec-CNN provides a much higher accuracy compared with onehot-CNN for all the values of k. We also show that loc2vec embedding improves the accuracy of about 40% on the average compared to onehot representation.

We also show that low-order loc2vec-CNN models worked as well or better than high-order ones, and better than both onehot-CNN and O(k) Markov models. Figure 5 shows that O(k) Markov is better than onehot-CNN only when k is small. It also shows that increasing the value of k in both loc2vec-CNN and onehot-CNN provides a smaller decreasing in the accuracy compared to the O(k) Markov model which exhibits a very high decreasing in the accuracy. For example, with O(k) Markov, the average accuracy decreases by more than 35% between the two values of $k = 8$ and $k = 64$ whereas with our models onehot-CNN and loc2vec-CNN, it only decreases by less than 9% and 4% respectively for the same values of k.

This means that CNN models are able to keep somewhat the accuracy even with a high order of context k due to its efficiency in learning patterns contrary to traditional mobility

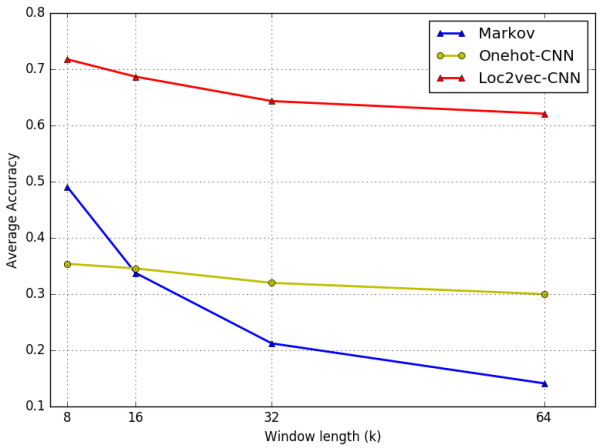

Fig. 5: Comparison of loc2vec-CNN, onehot-CNN and O(k) Markov, according to the average accuracy metric.

prediction models such as O(k) Markovian which are based on examining the history of sequence to predict the next location.

In Figure 6, we plot comparison results of 20 users with a context length $k = 8$, a sequence length s corresponding to one day, and a loc2vec embedding vector size $m = 10$. We show that the accuracy varies from a user to another for models loc2vec-CNN, onehot-CNN, and O(k) Markov. The results show that loc2vec-CNN provides the best accuracy compared to the other models for almost all considered users. Only in four cases, O(k) Markov provided slightly better accuracy compared to loc2vec-CNN.

In Figure 7, we compare the average accuracy of five existing CNN models by training the whole network from scratch, and by fine-tuning these models using transfer learning from existing popular pre-trained models. We considered the following models: SqueezeNet [35] (the model used to obtain the previous results in this paper), Inception [33], AlexNet [31], DenseNet [36] and ResNet [34]. We show that starting from an already trained CNN model provides a much higher accuracy compared to the training from scratch. We show that the average accuracy obtained with all the five considered CNN models increases with fine-tuning. The difference between a model trained from scratch and another one taking advantage of transfer learning after fine-tuning exceeds 7% for most considered models.

V. Conclusions

We have presented and evaluated several models for next location prediction using a subset of real mobility traces. In contrast to most existing proposals, we proposed to make use of modern machine learning techniques based on deep neural networks. We have proposed the use of Convolutional Neural Networks for which we enhanced the representation of input data by the use of embedding techniques. We have explicitly derived a new location embedding technique which we called loc2vec to enhance the quality of input location representations. Our loc2vec embedding technique improves the representation of locations by encoding close locations in mobility sequences in a way that makes their loc2vec representations also close after the embedding. We enhanced the performance of our CNN models that are based on loc2vec embedding with the use of transfer learning which allows us to take advantage of pre-trained CNN networks which we fine-tuned on our location prediction application domain. We evaluated the performance of our proposals on real mobility datasets and showed that the combination of loc2vec, CNN, and the use of transfer learning from existing CNN model provide the best results compared to popular state of the art prediction techniques relying on Markovian models.

References

[1] Logesh Ravi and Subramaniyaswamy Vairavasundaram. A collaborative location based travel recommendation system through enhanced rating prediction for the group of users. *Computational intelligence and neuroscience*, 2016:7, 2016.

[2] Anastasios Noulas, Salvatore Scellato, Neal Lathia, and Cecilia Mascolo. A random walk around the city: New venue recommendation in location-based social networks. In *Privacy, security, risk and trust (PASSAT), 2012 international conference on and 2012 international confernece on social computing (socialcom)*, pages 144–153. Ieee, 2012.

[3] Alicia Rodriguez-Carrion, Carlos Garcia-Rubio, Celeste Campo, Alberto Cortés-Martín, Estrella Garcia-Lozano, and Patricia Noriega-Vivas. Study of lz-based location prediction and its application to transportation recommender systems. *Sensors*, 12(6):7496–7517, 2012.

[4] Lauri Aalto, Nicklas Göthlin, Jani Korhonen, and Timo Ojala. Bluetooth and wap push based location-aware mobile advertising system. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 49–58. ACM, 2004.

[5] Natalia Marmasse and Chris Schmandt. Location-aware information delivery withcommotion. In *International Symposium on Handheld and Ubiquitous Computing*, pages 157–171. Springer, 2000.

[6] James Scott, AJ Bernheim Brush, John Krumm, Brian Meyers, Michael Hazas, Stephen Hodges, and Nicolas Villar. Preheat: controlling home heating using occupancy prediction. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 281–290. ACM, 2011.

[7] Marta C Gonzalez, Cesar A Hidalgo, and A-L Barabasi. Understanding individual human mobility patterns. *arXiv preprint arXiv:0806.1256*, 2008.

[8] Libo Song, David Kotz, Ravi Jain, and Xiaoning He. Evaluating location predictors with extensive wi-fi mobility data. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1414–1424. IEEE, 2004.

[9] Libo Song, David Kotz, Ravi Jain, and Xiaoning He. Evaluating next-cell predictors with extensive wi-fi mobility data. *IEEE Transactions on Mobile Computing*, 5(12):1633–1649, 2006.

[10] Akinori Asahara, Kishiko Maruyama, Akiko Sato, and Kouichi Seto. Pedestrian-movement prediction based on mixed markov-chain model. In *Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 25–33. ACM, 2011.

[11] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. Next place prediction using mobility markov chains. In *Proceedings of the First Workshop on Measurement, Privacy, and Mobility*, page 3. ACM, 2012.

[12] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. Show me how you move and i will tell you who you are. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*, pages 34–41. ACM, 2010.

[13] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. Predicting the next location: A recurrent model with spatial and temporal contexts. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 194–200. AAAI Press, 2016.

[14] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

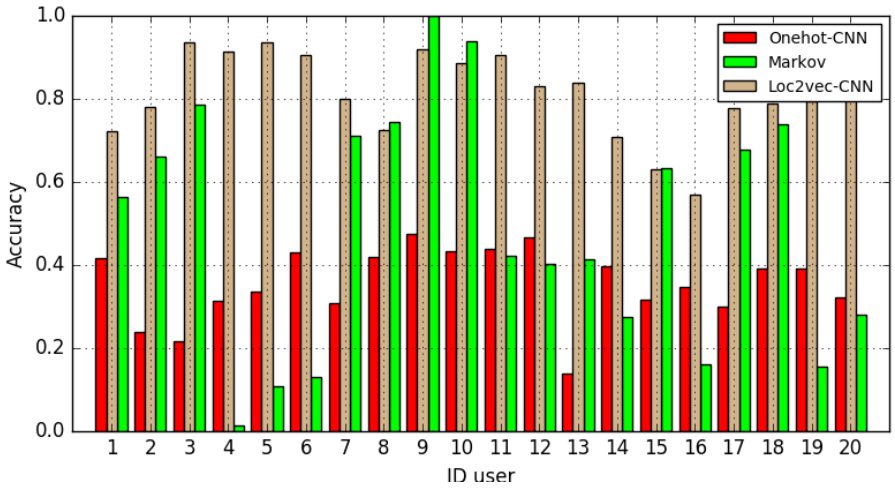

Fig. 6: Comparison of loc2vec-CNN, onehot-CNN and O(k) Markov according to the accuracy metric for every user.

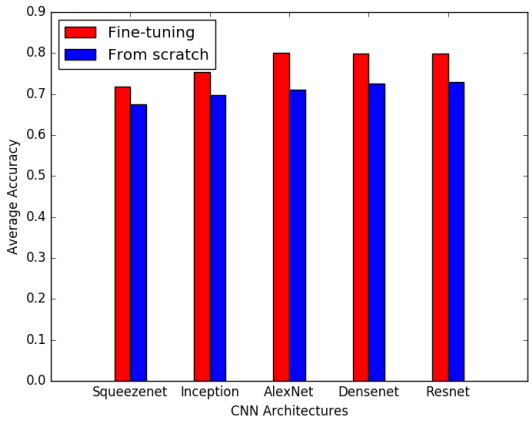

Fig. 7: Effect of pre-training on the performance of loc2vec-CNN.

[15] Fan Wu, Kun Fu, Yang Wang, Zhibin Xiao, and Xingyu Fu. A spatial-temporal-semantic neural network algorithm for location prediction on moving objects. *Algorithms*, 10(2):37, 2017.

[16] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[17] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[18] Songqing Yue. Imbalanced malware images classification: a cnn based approach. *arXiv preprint arXiv:1708.08042*, 2017.

[19] Sohaib Laraba, Mohammed Brahimi, Joëlle Tilmanne, and Thierry Dutoit. 3d skeleton-based action recognition by representing motion capture sequences as 2d-rgb images. *Computer Animation and Virtual Worlds*, 28(3-4):e1782, 2017.

[20] Thao Le Minh, Nakamasa Inoue, and Koichi Shinoda. A fine-to-coarse convolutional neural network for 3d human action recognition. *arXiv preprint arXiv:1805.11790*, 2018.

[21] Venkatesh Boddapati, Andrej Petef, Jim Rasmusson, and Lars Lundberg. Classifying environmental sounds using image recognition networks. *Procedia Computer Science*, 112:2048–2056, 2017.

[22] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[23] David Kotz and Kobby Essien. Analysis of a campus-wide wireless network. *Wireless Networks*, 11(1-2):115–133, 2005.

[24] Tristan Henderson, David Kotz, and Ilya Abyzov. The changing usage of a mature campus-wide wireless network. *Computer Networks*, 52(14):2690–2712, 2008.

[25] Paul Baumann, Wilhelm Kleiminger, and Silvia Santini. The influence of temporal and spatial features on the performance of next-place prediction algorithms. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 449–458. ACM, 2013.

[26] Monit Shah Singh, Vinaychandran Pondenkandath, Bo Zhou, Paul Lukowicz, and Marcus Liwickit. Transforming sensor data to the image domain for deep learningan application to footstep detection. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 2665–2672. IEEE, 2017.

[27] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, may 2015.

[28] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[29] Jose Camacho-Collados and Mohammad Taher Pilehvar. From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research*, 63:743–788, 2018.

[30] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[32] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions.

[33] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[35] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[36] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks.

[37] David Kotz, Tristan Henderson, Ilya Abyzov, and Jihwang Yeo. CRAWDAD dataset dartmouth/campus (v. 2009-09-09). Downloaded from http://crawdad.org/dartmouth/campus/20090909, September 2009.

[38] Radim Rehurek and Petr Sojka. Software framework for topic modelling with large corpora. In *In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Citeseer, 2010.