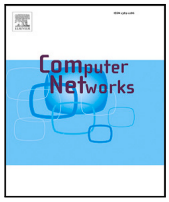
Computer Networks 252 (2024) 110657

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Next-cell and mobility prediction in new generation cellular systems based on convolutional neural networks and encoding mobility data as images

Peppino Fazio [a,b,*], Miralem Mehic [b,c], Miroslav Voznak [b]

[a] Department of Molecular Sciences and Nanosystems, Ca' Foscari University, Via Torino 155, Mestre, 30172, Italy
[b] VSB – Technical University of Ostrava, 17. listopadu 2172/15, Ostrava, 70800, Czechia,
[c] Department of Telecommunications, Faculty of Electrical Engineering, University of Sarajevo, Zmaja od Bosne bb, Sarajevo, 71000, Bosnia and Herzegovina

ARTICLE INFO

Keywords:
CNN
Data-2-image conversion
Machine learning
Mobility classification
Pattern prediction
5G

ABSTRACT

Mobility prediction has been a popular research topic for many decades. With the advent of new generation technologies (5G and beyond) and smaller coverage cells, hand-over operations have become more frequent. Cellular system companies are therefore taking increasing interest in using the available predictive information on node movements to optimize and manage their bandwidth resources. In particular, the main challenging scope of our contribution consists in solving the issue of reliable next-cell prediction, aimed to call dropping probability minimization. In addition, our proposal is based on the innovative concept of mobility data to image encoding. The scheme is able to a-priori determine the next visited cells during host movements by applying a convolutional neural approach to mobility images. The power of machine learning is used to advantage, and highly accurate image classification is achieved for mobility prediction. We performed numerous simulation campaigns related to next-cell prediction in mobile cellular environments, obtaining very satisfactory results by the application of convolutional neural networks, which have an impressive history of effectiveness with image classification problems. The trained network has been associated to each coverage cell and the prediction accuracy has been evaluated.

1. Introduction

In the last few decades, mobile computing has rapidly evolved to deliver higher transmission speeds and lower latencies, which are the compelling features of 5G and beyond architectures [1]. The attainable Quality of Service (QoS) and Quality of Experience (QoE) have also grown at a greater than linear rate. Integration of these standards with predictive approaches to provide high-quality assistance with resource management, especially if network nodes move within a cellular network, is therefore highly coveted [2]. Much attention has also been devoted to Machine and Deep Learning (ML, DL) applications to improve their recognition, classification, and prediction accuracy of the main features of several processes in very different research areas [3–5].

The target of our proposal regards the new generation cellular systems, such as 5G and 6G, where the hand-over events are becoming more and more frequent due to the smaller coverage area. Especially in the future 6G networks [1], where the ML/DL approaches will be integrated with the architecture, the overall system will be aided to a-priori know mobile host movements to pre-reserve the needed amount of bandwidth (or another kind of resource blocks) when the users will arrive in the main locations. We want to give a valid contribution and possible solution to the reliable next-cell prediction issue aimed at Call Dropping Probability (CDP) minimization.

In contrast to the vast majority of existing works (which implement and evaluate a dedicated ML structure to predict a specific process), we propose a new predictive scheme based on the strength of Convolutional Neural Networks (CNNs) [6] in image recognition and classification. However, we do not propose any new CNN layering but rather a completely innovative method for encoding and predicting (in terms of following visited cells) images in cellular mobility using CNN image classification. This approach's main aim is to demonstrate how the high accuracy of CNNs [7] can be applied to next-cell and mobility prediction. To our knowledge, no mobility approaches can obtain an accuracy level greater than 90% [2,8] in cellular mobile networking. First, we present a completely innovative mobility image encoding algorithm that accounts for the obtained image entropy (which we consider an indicator of the mobility information level contained in the image) [9] and *meaningfully* encodes images to train a CNN adequately. We created mobility traces using several maps extracted from Open Street Map (OSM) [10] and simulated real mobility trajectories with

* Corresponding author at: Department of Molecular Sciences and Nanosystems, Ca' Foscari University, Via Torino 155, Mestre, 30172, Italy.
E-mail address: peppino.fazio@unive.it (P. Fazio).

https://doi.org/10.1016/j.comnet.2024.110657
Received 22 August 2023; Received in revised form 20 June 2024; Accepted 16 July 2024
Available online 25 July 2024

the Simulation of Urban MObility (SUMO) [11]. Cellular coverage for the maps was then added (modeled with Voronoi Tessellation [12]), and the CNN was trained in the hand-over event function. The effects of average coverage radius (varying the number of tessellation generator points) were also included. Mobility was then sampled, and a mobility points subset was periodically allocated. Images were obtained using a specific encoding algorithm. These images were then used as a training set for the CNN to learn the specific mobility features contained within the images. We highlight that no new CNN layering is proposed; we are only interested in demonstrating the potential of exploiting the strength of CNNs with image classification (already proposed by us in [13] for mobile scenario) for mobility prediction. The main contributions of this study are:

- Adapting a Mobility-2-Image encoding algorithm proposed in [13] for geographical areas classification to predict the future positions of mobile hosts: in particular, in this new proposal, the CNN can predict the future cell that a mobile host (whose coordinates have been converted into an image) will visit (if any);
- Using CNN's accuracy for image recognition without needing new CNN layering. Node mobility is treated as an image set which can be classified and predicted;
- An analysis of the accuracy of a CNN's functioning in encoding parameters such as image size, sampling frequency, encoding algorithm approach, etc.

We assume that mobility data can be collected without the availability of a GPS device at mobile nodes. Base stations can store mobility coordinates through several methods, for example, Angle-of-Arrival (AoA), Proximity, Trilateration, and Time Difference of Arrival (TDoA) [14].

The remainder of the paper is organized as follows: Section 2 introduces recent works on mobility sampling, mobility prediction, and image classification; Section 3 describes the main proposed concept, divided into five subsections. Section 4 provides a detailed description of the achievable results using the proposed algorithm; finally, Section 5 concludes the paper and comments on the main advantages of the proposed approach.

2. Related work

This section reviews the most recent and significant works on mobility prediction in cellular systems and image classification. These are the two main topics on which our proposal is based, and, to the best of our knowledge, they have never been combined before to implement a predictive model. We first briefly overview the main contributions in those areas of interest, then propose a new way to combine image classification with cellular prediction. The main goal of this section is to show how many different models have been proposed instead of thinking to standardize a unique approach. In addition, it should be underlined that the next-cell prediction serves as a desirable tool for network operators to optimize network resource allocation. With the deployment of femto/nano-cells in 5G architecture, hand-over events have become more frequent, and next-cell prediction is, therefore, a popular research area.

2.1. The heterogeneity of the existing mobility prediction models

Integrating prediction features into future cellular systems (5G and beyond) is highly desirable and offers optimizations for bandwidth management and resource allocation, as well as enhancement of QoS and QoE. The next-cell prediction problem has been studied for decades and documented in several related articles. In [2,8], and [15], the authors surveyed the main existing contributions, citing interesting details of methodologies and approaches. These works, although 4–7 years old, described the majority of predictive models proposed over many years (e.g., Kalman filters, Markov chains, Neural Networks, Autoregressive ML and DL, etc.), showing a colossal heterogeneity between the proposed models, with a general attainable accuracy below 90%. The trend has been the same in the last few years. The following articles are examples of the most recent continuous proposals of new predictive models. The authors of [16] proposed an attentional Markov model that extrapolated long-term correlations from historical trajectories in specific contexts. A detailed and large dataset was applied, significantly improving classic ML approaches (Hidden Markov Model, Recurrent Neural Network, Friendship, Mobility, etc.) in accuracy and execution time. In [17], the authors propose a new traffic prediction algorithm based on the Butterworth filter, CNN, and LSTM, also introducing a new approach of frequency domain analysis to extract the low-frequency component of the considered "signals". The dedicated CNN–LSTM prediction models are then used to represent the final value. The approach is entirely based on the Pytorch experimental environment, obtaining an enhancement of about 25% concerning the existing time series predictors.

The authors of [18] address the issue of mobility and traffic prediction by proposing a spatio-temporal graph attention model: the proposed scheme can quantify the strength of connections between nodes by evaluating the similarity matrix and constructing an adjacency matrix. The main novelty of the proposal consists of the fusion of spatio-temporal features in traffic data by combining the graph neural network with a temporal attention mechanism. Based on a real dataset (extracted from an actual service provider database), the authors show that their idea can outperform existing methods in terms of prediction accuracy.

In [19], the authors underline the importance of regional prediction and edge computing. By relocating the requested content to the edge in the Internet of Vehicles (IoV), edge caching is anticipated to effectively meet the low latency and high-reliability demands of IoV users for multimedia services. The authors proposed a mobility-aware proactive edge caching scheme (MSTPS), incorporating spatial and temporal predictions of vehicle movements for content deployment and scheduling. The scheme adapts to mobility uncertainties by learning vehicles' travel preferences and grouping users with similar travel patterns. To address the dynamic and unpredictable nature of the IoV, the authors designed a system recovery strategy to prevent degradation of the scheme due to prediction failures. Real taxi datasets and urban scenarios have been used, showing that MSTPS is able to improve the cache hit ratio, reducing access latency, and lowering the cache redundancy ratio.

Another example of a deep research about regional prediction is represented by [20]. The authors focused on regional traffic flow forecasting, crucial for the implementation of intelligent transportation systems. The authors defined another dedicated Graph Convolutional Neural network (MGCN) to analyze and express the spatial correlation of traffic flow across different dimensions. Then, an LSTM was constructed to capture temporal features, and an Embedding layer was added to embed temporal periodic features of traffic flow. The performance was tested using the public dataset PeMSD4 [21] and corresponding weather data. Compared to other prediction models, the final constructed prediction model demonstrated reduced prediction errors, indicating that the MGCN offers superior traffic flow prediction performance.

2.2. Image-data classification with neural networks

ML and CNNs have been used extensively in image processing and recognition for several years. As evidence, the work in [22] overviews the different fields in which CNNs have been employed to recognize (and/or restore) images: from animals to hand-made objects, from people to plants, from buildings to labeled objects, etc. Until today, the common features of CNN applications consist in the used datasets, which always contain real-world objects (generally speaking, image data), and in the exploitation of the power of convolutional layers,

Fig. 1. The general approach considered for obtaining the starting mobility data and coverage cells, from Open Street Map and SUMO.

which can discover hidden features in input images, thereby allowing subsequent layers to output correct classified values. Very high accuracy in recognizing complex/textured features is reached, outperforming classic approaches as demonstrated, for example, in [23], where the authors examined the problem of even rebuilding and restoring degraded images (blurred, noisy, or distorted) after classification. The authors tested and compared a classic CNN and a CNN with prior filtering to remove image degradation. The experiment showed poorer image classification performance in the CNN with degraded images and unsatisfactory performance from the degradation removal algorithms, which could not improve the results. The study in [24] represents another recent and valid contribution, where the authors investigated the compressed-domain image classification, considering the desired image format before encoding (the reconstruction step was bypassed). The authors tested their proposed method on large datasets (as Coil-100 [25]), obtaining satisfactory performance and resilience to noise. The authors of [26] proposed the Visual Geometry Group 16 (VGG16) model for classifying images into two additional categories instead of performing feature extraction or segmentation. VGG16 offers an accuracy of 99%, and images are further categorized into sub-categories. In [27], an interesting proposal to deal with synthetic images is given: the authors deal with a detailed approach for extracting features (similarity, stability, transferability) from images and encode them through a two stage (generation and distillation) federated learning approach. From the discussion above, it is evident that CNNs have been mostly used for classifying only image-data, while the classification of synthetic and/or non-image-data images represents a very young approach to train and validate convolutional models.

Unlike the works mentioned above, we present a new approach to mobility prediction through image classification without any new CNN layering. A straightforward CNN structure is considered, as the one first proposed by LeCun in [28]. Contrary to most proposals, we adapted the model for compatibility with CNNs by encoding mobility representations as images.

3. Next-cell prediction with a CNN for image classification

This section describes the components of our proposed model, starting with a coverage model, continuing with mobility to image conversion, and finishing with a predictive algorithm. We underline that the predictive algorithm is the minor contribution of this paper: the attention is focused on suggesting, for the first time, how to use CNNs (used for image classification/recognition) in mobile networking by converting mobility data to images hence having the possibility to make predictions on users movements. Just for sake of simplicity and for introducing the operative approach proposed in this contribution, Fig. 1 is shown. The real mobility map (on the left side) in the OSM format can be imported in SUMO (on the rights side), which is able to generate real mobility traces taking into account real features (traffic lights, maximum speed limits, crosses, downtowns, etc.); the yellow boxes on the right side represent the moving vehicles. SUMO is able to store mobility in GPS Decimal Degrees (D.D°) format, and an overlay cellular architecture can be easily implemented, in order to consider the real hand-over events. The overlay cellular network consist of 7 clusters of 7 cells, covering the Geographic Area (GA) of interest. These concepts will be deeply explained throughout the paper.

The following sections discuss the main results of this approach, and we emphasize again that no new CNN structure is proposed; instead, an entirely new approach to next-cell prediction can exploit CNN's accurate image classification capability. Our approach is general and independent of any specific coverage or a vehicle's movement behavior (pedestrian, car, mobile sensor, etc.).

3.1. The general model for creating a suitable dataset

Let us consider a rectangular, two-dimensional GA. We describe a two-dimensional (rectangular) mobility environment for simplicity, although the approach could be easily adapted to describe three dimensions. The GA is characterized by its dimensions GA_x and GA_y, area $GA_A = GA_x \cdot GA_y$, and a set of contained roads $GA_R = \{r_1, \ldots, r_m\}$, where $|GA_R| = m$ and $m > 0$. User mobility is possible only on each $r_k \in GA_R$. Therefore, considering a set of mobile hosts $MH = \{mh_1, \ldots, mh_n\}$, where $|MH| = n$, $n > 0$, and for a certain time t:

$$mh_j(t) \in r_k, \tag{1}$$

where $t \in [t^0_{jk}, t^1_{jk}]$ (the interval during which mh_j moves on r_k), $mh_j \in MH$, $r_k \in GA_R$. Thus, a mobile host mh_j will traverse a set of l_j roads $R_j = \{r^j_1, \ldots, r^j_{l_j}\}$ during an active session (data connection) of duration T_j.

The model can be extended by adding a set of coverage cells (for now, we are not concerned whether they belong to 3G, 4G or 5G base stations or femto/pico cells) $GA_C = \{c_1, \ldots, c_p\}$ characterized by a shape, a number of available channels (managed by CDMA, NOMA, W-CDMA, or TD-SCDMA [29]) and a coverage radius (if the shape is regular) or an average coverage radius (if the shape is irregular, as shown later). Eq. (1) can be rewritten as:

$$mh_{j,k}(t) \in c_i, \tag{2}$$

meaning that, at time t, user mh_j travels on r_k in the coverage of cell c_i. During T_j, we can thus observe mh_j movements from both R_j and $C_j = \{c^j_1, \ldots, c^j_{p_j}\}$ points of view, where $|C_j| = p_j$ and represents the number of cells visited by mh_j in T_j, and $C_j \subseteq GA_C$.

Fig. 2 illustrates an example of a GA (extracted from a map of the center of Los Angeles, USA) with an area GA_A of approximately 0.4 km^2, and a set of roads GA_R with $m = 27$ (street numbers were omitted from the map the figure for clearer readability). A generic

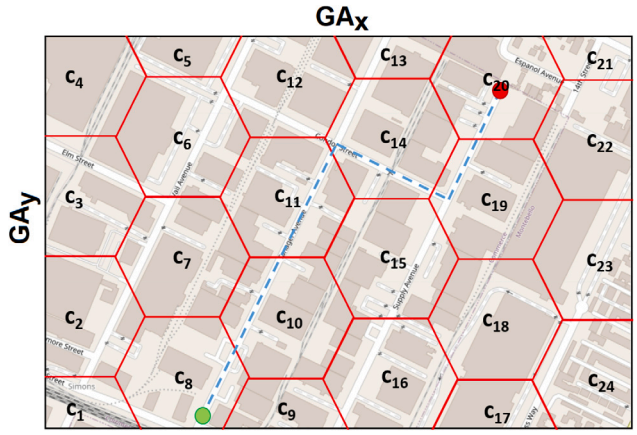


Fig. 2. Example of a GA with regular coverage, $GA_x \approx 782$ m, $GA_y \approx 512$ m, $GA_C = \{c_1, \ldots, c_{24}\}$, $GA_R = \{r_1, \ldots, r_{27}\}$, $C_j = \{c_8^j, c_{10}^j, c_{11}^j, c_{14}^j, c_{19}^j, c_{20}^j\}$, $R_j = \{r_7^j, r_9^j, r_{10}^j\}$.

mobile host, mh_j travels from the green dot to the red dot, traversing the set of roads $R_j = \{r_7^j, r_9^j, r_{10}^j\}$ (Tanager Avenue, Condor Street, and Supply Avenue, respectively). Cellular coverage (for simplicity, regular hexagonal shapes) was overlaid on the GA, where $p = 24$ (coverage radius of 65 m), and the route's trajectory (mh_j) forces the node to hand over between cells $C_j = \{c_8^j, c_{10}^j, c_{11}^j, c_{14}^j, c_{19}^j, c_{20}^j\}$ (indicated by the blue dotted line). For the general case of irregular cellular coverage, Voronoi's tessellation is considered the best solution [12,30] to describe the distribution of cell coverage signal strength between boundaries. Once we define the boundaries of a two-dimensional GA, we can set the number of Voronoi regions p and their related generator points to $GA_P = \{gp_1, gp_2, \ldots, gp_p\}$ such that $gp_i \neq gp_j$ for $i \neq j$, and $gp_i \in \mathbb{R}^2$. Given gp_i, we can also define the Voronoi region related to cell $c_i \in GA_C$, which satisfies:

$$A(c_i) = \{P \in \mathbb{R}^2 | \, \|P - gp_i\| \leq \|P - gp_j\|, \forall j \neq i\}, \tag{3}$$

where $A(c_i)$ is the coverage region of c_i in $\mathbb{R}^2$ and $\|\cdot\|$ is the Euclidean distance. All the coverage regions of a Voronoi tessellation in $\mathbb{R}^2$ are connected and convex, but we also have to consider a Voronoi region bounded to the GA. Therefore,

$$A^*(c_i) = A(c_i) \cap GA, \quad GA = \bigcup_{i=1}^{p} A^*(c_i). \tag{4}$$

In addition to the generator points, defining edges in Voronoi coverage is important. The edge between $A^*(c_i)$ and $A^*(c_j)$ is defined as $e(gp_i, gp_j) = A^*(c_i) \cap A^*(c_j)$, and if $e(gp_i, gp_j) \neq 0$, then c_i and c_j are adjacent. For each point $P \in e(gp_i, gp_j) \neq 0$, we have $\|P - gp_i\| = \|P - gp_j\|$.

For our purposes, each coverage cell (Voronoi area) can, therefore, be considered a polygon (with an irregular shape) bounded by a certain number of edges. Fig. 3 illustrates an example extracted from a complete Voronoi tessellation in which six generator points are defined, and the area generated by gp_3 has five edges (blue numbers) shared with neighboring cells. Generally, we can say that each cell $c_i \in GA_C$ has q_i edges and, therefore, q_i possible hand-in/hand-out directions. Without loss of generality, we assume that Eq. (1) remains valid during the hand-over event (users change the coverage cell but not the road on which they are contextually moving). We want to highlight that the management of the handover procedure (in terms of signaling between devices) is beyond the scope of the current study.

3.2. Dataset creation: Mobile host

Here, we propose a new method for preparing the mobility dataset. Let us set a sampling frequency of f_s (mobility positions are sampled

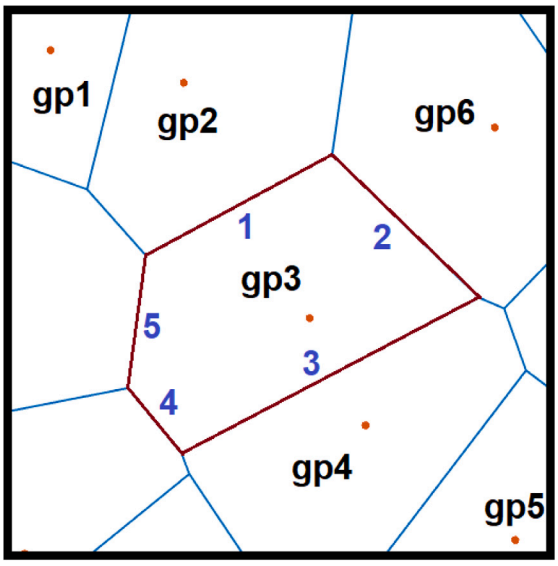


Fig. 3. Example of a part of a Voronoi tessellation.

f_s times per second) and a coordinate reference point $P_0 = (0, 0)$ in the GA. For each $mh_j \in MH$, a mobility dataset can then be created with the following fields in each tuple:

- mh_j – the IDentifier (ID) of the mobile host, assigned by the current coverage cell;
- $curr_{time}$ – the mobility sampling time related to the coordinates storage event;
- pos_x – the x-coordinate of mh_j at time $curr_{time}$, referring to P_0 (this can also be interpreted as the latitude value if the Global Navigation Satellite System – GNSS – is used);
- pos_y – the y-coordinate of mh_j at time $curr_{time}$, referring to P_0 (this can also be interpreted as the longitude value if GNSS is used);
- $road_k$ – the $r_k \in GA_R$ which mh_j traverses at time $curr_{time}$ (referred in Eq. (1));
- $cell_i$ – the $c_i \in GA_C$ which covers mh_j at time $curr_{time}$ by $A^*(c_i)$ (referred in Eq. (2));
- $prev_{cell}$: the $c_v \in GA_C$ which covered mh_j at time $curr_{time} - \frac{1}{f_s}$ by $A^*(c_v)$; this is the field $cell_i$ at the previous sampling time.

Let us assume that during the training phase, each moving $mh_j \in MH$ periodically (each $1/f_s$ seconds) creates an updated tuple according to Algorithm 1 and appends it to the current dataset (we refer to this as the *historical* period). The algorithm commences when the connection of mh_j becomes active. Then, if a hand-over event occurs, the most recent version of the dataset is sent to the *old* covering cell before the hand-over procedure completes. The dataset is also sent to the current covering cell when the connection becomes inactive (the connection ends). Each $c_i \in GA_C$ builds its historical dataset by storing the tuples received by all $mh_j \in MH$. Note that the algorithm is executed only during the *historical* period, meaning only when the historical dataset needs to be created and shared in the system (to correctly train the CNN, as shown later). Regarding the computational complexity of Algorithm 1, the algorithm accepts current time t_0 and sampling frequency f_s as input for evaluating the time instants when mobility should be sampled. Then, it initializes all the required variables and data structures (these operations are performed in real-time). The variable $prev_{cell}$ is set to -1 to indicate that the connection begins in the current cell and that no previous hand-over events are present. The mobile host also receives its unique ID directly from the

Algorithm 1: Dataset creation from mh_j to cell c_i

Result: Creates tuples from mh_j mobility and transmits them to the current coverage cell c_i.

input: f_s, t_0; start init:
set T_s=1/f_s, $ID = acquire_{ID}(j)$, $active$=true,
$current_{time} = t_0$, $c_i = current_{cell}()$, $prev_{cell} = -1$,
$current_{mode} = historical$, $dataset = \emptyset$;
end init;
while $(active \wedge current_{mode} == historical)$ **do**
  if *is_sampling_time*(T_s) **then**
    $create\ tuple = (ID, current_{time}, position(pos_x),$
    $position(pos_y), road(), c_i, prev_{cell})$;
    $append(tuple, dataset)$;
    $current_{time} = current_{time} + T_s$
  if *hand_over reserved* **then**
    $prev_{cell} = c_i$;
    $c_i = next_{cell}()$;
    $create\ tuple = (ID, current_{time}, position(pos_x),$
    $position(pos_y), road(), c_i, prev_{cell})$;
    $append(tuple, dataset)$;
    $send_{to}(prev_{cell}, dataset)$;
    $empty(dataset)$;
    $append(tuple, dataset)$;
  if *connection_stopped*() **then**
    $create\ tuple(ID, connection_{ended})$;
    $append(tuple, dataset)$;
    $send_{to}(c_i, dataset)$;
    active=false;
  if *historical timeout reached* **then**
    $current_{mode}$=stop;

current c_i through the function $acquire_{ID}(j)$. When the mobile host mh_j connects, it creates the appropriate tuples (containing the mobility samples) every T_s seconds. Suppose an active connection lasts for Call Holding Time seconds [31] under the coverage of c_i ($CHT_{i,j}$). In that case, the moving node will have $CHT_{i,j}/T_s = k_1$ sampling and storage iterations, plus a final tuple which is created immediately before it exits the coverage of the current cell c_i (this last operation is performed only if mh_j leaves c_i along its itinerary, giving c_i information about the hand-out direction and the new coverage cell information about the arrival edge and incoming direction). Then, if hand-over or the connection ceases, the created dataset is transmitted to c_i using the $send_{to}$ function. If we assume that the dataset created for cell c_i is composed of k_2 tuples, the code executed in the first If is $O(k_2)$, in the second it is $O(k_2+1)$, and in the third it is $O(k_2)$. The overall complexity is therefore polynomial and expressed as $O[k_1 \cdot (k_2 + k_2 + 1 + k_2)] \approx O(k_1 \cdot k_2)$, neglecting the constant terms.

3.3. Dataset creation: Pre-processing infrastructure to prepare the training set

Referring to a coverage cell c_i, we can define several STATUSES for a mobile host $mh_j \in MH$ currently covered by c_i:

- (a) $BORN_{ON} - mh_j$ starts its connection in c_i without any previous hand-over event and then ends its connection in the current c_i;
- (b) $BORN_{OUT} - mh_j$ starts its connection in c_i without any previous hand-over event and then leaves the current c_i with a hand-out event (in the direction OUT);
- (c) $HAND_{IN-OUT} - mh_j$ arrives in c_i with a previously active connection, and therefore a hand-over event occurs (from the direction IN); mh_j also leaves c_i and enters an adjacent cell (in the direction OUT);

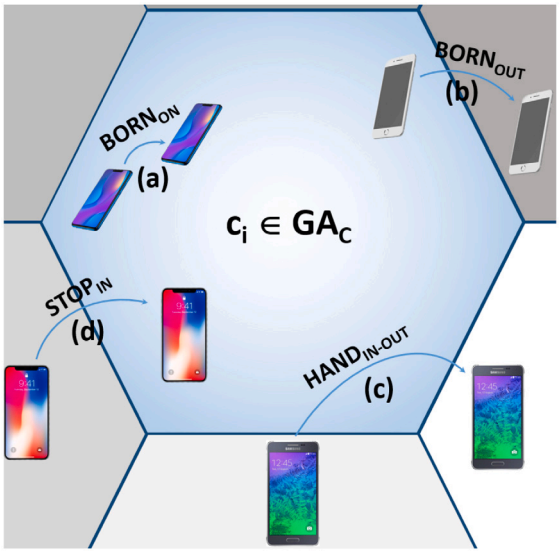


Fig. 4. Possible hand-over statuses for a mobile host which contacts c_i.

- (d) $STOP_{IN} - mh_j$ arrives in c_i (from the direction IN) with an active connection after a hand-over event and ends the connection without handing over c_i.

Fig. 4 summarizes the possible hand-over statuses for several mobile hosts covered by c_i and the number of total statuses depend on the number of possible hand-over directions.

Algorithm 2 refers to the statuses for creating opportune sub-datasets (we can say that mobility data is grouped into several classes according to the possible statuses). Before proceeding, we must also define each cell's possible hand-over directions.

Given $c_i \in GA_C$ with q_i edges, each $mh_j \in MH$ may enter/exit c_i from/to directions $D_i = \{d_1, \ldots, d_{q_i}\}$. The possible defined mobility states therefore become $BORN_{ON}$, $BORN_{d_{out}}$, $HAND_{d_{in},d_{out}}$, $STOP_{d_{in}}$, where d_{in}, $d_{out} \in D_i$. Directions refer directly to the q_i edges of cell c_i, and a hand-off event occurs when

$$mh_{j,k}(t) \in c_i \wedge mh_{j,k}(t + \frac{1}{f_s}) \in c_j,\ e(gp_i, gp_j) \neq 0, i \neq j. \quad (5)$$

For regular coverage, the direction sets are the same thus $D_i = D \ \ \forall c_i \in GA_C$. Regarding the infrastructure, each cell $c_i \in GA_C$ acts according to Algorithm 2.

The algorithm for c_i receives the value of q_i (i.e., the number of possible hand-in/hand-out directions) as input and then initializes historical mode, where cell c_i expects to receive tuples from the mobile hosts under its coverage, and initializes the datasets with the relevant mobile host statuses ($BORN_{ON}$, $BORN_{OUT}$, $STOP_{in}$, and $HAND_{in,out}$). When a set of tuples is received, it is ordered according to the IDs of the sender mh_j and stored in a temporary database. When the historical period finishes (it depends on the architecture and the CNN's training method), the algorithm begins populating the status datasets by considering all possible conditions ∀ ID:

- The $prev_{cell}$ field of the first tuple is equal to −1 (born under the coverage of c_i), and the last is a $connection_{ended}$ tuple, meaning that the ID mobile host mobility referred to c_i belongs the $BORN_{ON}$ status.
- The $prev_{cell}$ field of the first tuple is equal to −1, and the last is not a $connection_{ended}$ tuple, meaning that the ID mobile host mobility referred to c_i starts under the coverage of c_i and is then handed out in the *handout* direction of the edge shared with the next cell. Thus, the related status is $BORN_{handout}$.

Algorithm 2: Tuples storage and post-processing required to create the classified training dataset for cell c_i

Result: Stores the received tuples during the historical period, then classifies the different received traces.

input: q_i;
start init:
set $current_{mode}$=historical, $BORN_{ON} = \emptyset$;
set $dataset_{db} = \emptyset$;
for $in = 1;\ in++;\ in \leq q_i$ **do**
  $OUT = in$; $BORN_{OUT} = \emptyset$; $STOP_{in} = \emptyset$;
  for $out = 1;\ out++;\ out \leq q_i$ **do**
    $HAND_{in-out} = \emptyset$;
end init; **while** $current_{mode} == historical$ **do**
  if $tuple_{dataset}$ *is received* **then**
    $addOrdered(tuple_{dataset}, sender_{ID}, dataset_{db})$;
  if *historical timeout reached* **then**
    $current_{mode}$=pre-processing;
$min_{ID} = minimum_{ID}(dataset_{db})$; $max_{ID} = maximum_{ID}(dataset_{db})$; **for** $ID = min_{ID};\ ID++;\ ID \leq max_{ID}$ **do**
  if $exists(ID, dataset_{db})$ **then**
    $Destination_{dataset} = \emptyset$; $first_{tuple} = first(ID, dataset_{db})$;
    $last_{tuple} = last(ID, dataset_{db})$;
    if $\{(first_{tuple}.prev_{cell} == -1) \wedge (last_{tuple} == connection_{ended})\}$ **then**
      $Destination_{dataset} = BORN_{ON}$;
    else if $\{(first_{tuple}.prev_{cell} == -1) \wedge (last_{tuple} \neq connection_{ended})\}$ **then**
      $handout = direction(last_{tuple}.cell_i)$;
      $Destination_{dataset} = BORN_{handout}$;
    else if $\{(first_{tuple}.prev_{cell} \neq -1) \wedge (first_{tuple}.prev_{cell} \neq c_i) \wedge (last_{tuple} \neq connection_{ended})\}$ **then**
      $handin = direction(first_{tuple}.prev_{cell})$;
      $handout = direction(last_{tuple}.cell_i)$;
      $Destination_{dataset} = HAND_{handin-handout}$;
    else if $\{(first_{tuple}.prev_{cell} \neq -1) \wedge (first_{tuple}.prev_{cell} \neq c_i) \wedge (last_{tuple} == connection_{ended})\}$ **then**
      $handin = direction(first_{tuple}.prev_{cell})$;
      $Destination_{dataset} = STOP_{handin}$;
    $addTuples(dataset_{db}, ID, Destination_{dataset})$;

- The $prev_{cell}$ field of the first tuple is equal to neither -1 nor c_i, and the last is not a $connection_{ended}$ tuple, meaning that the ID mobile host arrives in the current cell from the *handin* direction of the edge shared with the cell in the $prev_{cell}$ field and is handed out in the *handout* direction of the edge shared with the cell in the $cell_i$ field of the last tuple. Thus, the related status is $HAND_{handin-handout}$.
- The $prev_{cell}$ field of the first tuple is equal to neither -1 nor c_i, and the last is a $connection_{ended}$ tuple, meaning that the ID mobile host arrives in the current cell from the *handin* direction of the edge shared with the cell in the $prev_{cell}$ field and the connection then ends. Thus, the related status is $STOP_{handin}$.

Once the status is determined for each mobile host ID, the related dataset is added to the dataset linked to the status. The four datasets (of the four states) thus contain mobility tuples describing the four conditions illustrated in Fig. 4. Regarding the computational complexity of Algorithm 2, the first nested *For* initialization cycles are $\theta(q_i^2)$, the code inside the second cycle (*while*) is instead $\theta(k_2)$ (we assume that the $tuple_{dataset}$ to be added has k_2 tuples). The cycle is repeated k_0 times, where k_0 represents the number of mobile hosts transmitting data. The final *For* cycle may then be considered a classification operation, repeated k_0 times (the number of valid mobile hosts whose tuples are present in the main database). Each iteration is $O(k_3)$, where k_3 is the overall number of tuples previously added to the database. The complexity of Algorithm 2 is expressed as $\theta(q_i^2) + \theta(k_2) + O(k_0 \cdot k_3) \approx O(k_0 \cdot k_3)$, with $q_i, k_2 \ll k_0$ and $q_i, k_2 \ll k_3$.

3.4. Conversion of mobility data into images: CNN training set creation and image recognition

This subsection introduces the main innovation behind the proposed method of converting categorized mobility data into images and subsequent recognition and prediction. In this manner, when a user moves into a c_i, the current coverage cell and its associated CNN can classify the moving user and thereby anticipate its future behavior. This is an extension of the approach proposed in [13], where the CNN can determine which is the geographical area in which a mobile host is moving.

To this aim, we assume that each $c_i \in GA_C$ can train and deploy a CNN, denoted by CNN_i. The related training set ts_i is composed of the data classified and contained in the created $BORN_{ON}$, $BORN_{OUT}$, $HAND_{IN-OUT}$, and $STOP_{IN}$ sub-datasets, but before they can be applied, we propose converting them into images and exploiting the high accuracy of image recognition in the CNNs.

Recalling that q_i represents the number of edges of c_i, we can derive the maximum number of sub-datasets created by Algorithm 2. Only one $BORN_{ON}$ sub-dataset (it does not depend on q_i), q_i $BORN_{OUT}$ sub-datasets (the OUT direction is related to q_i), q_i^2 $HAND_{IN-OUT}$ sub-datasets, and q_i $STOP_{IN}$ sub-datasets will be created, therefore c_i's maximum number of sub-datasets is:

$$MAX - SD_i = 1 + q_i + q_i^2 + q_i = q_i^2 + 2 \cdot q_i + 1 = (q_i + 1)^2 \tag{6}$$

This represents an upper-bound only because, given the set of roads covered, users will move according to Eqs. (1) and (2), and therefore the probability of considering a $MAX - SD$ number of datasets is very low (this will be illustrated in the numerical results). The main aim now is to convert the data contained in the $O[(q_i + 1)^2]$ sub-datasets into images to adequately train the CNN_i with the categorized images for image recognition. With the above considerations, for each tuple of mh_j belonging to ts_i, we can write

$$\{(pos_x, pos_y) \in A^*(c_i)\} \ \ \forall\ curr_{time}. \tag{7}$$

That is, each point of the mobility pattern of mh_j is bounded by the extension area of c_i (in two-dimensional scenarios, but it may be extended to three dimensions), and referring to the notation of Eq. (5), only the tuples for which $mh_{j,k}(t) \in c_i$ are considered.

To better understand the general idea, we want to stress that the four classes have been considered as mobility classes to train the CNN network (obviously taking into account the number of possible hand-in/hand-out directions). As illustrated in the work, for a $i-$th cell with q sides, we will have a maximum number of $MAX - SD_i$ classes. As mentioned above, during the historical phase (mobility data collection), each host that moves inside c_i is classified after its departure (hand-out) or service termination as belonging to one of the $MAX - SD_i$ classes. So, for those classes, the stored mobility coordinates (x and y) are grouped each $1/f_s$ second, and an image is built up with a proper encoding algorithm. So, each class contains "mobility images" regarding mobile hosts that have moved belonging to the particular class. At this point, when a certain number of image files has been obtained for each class (obviously, there are classes with no images because users did not move according to the "rules" of that class), the CNN model can be trained, to extract and learn all the features that are common to the users which moved in the "same" way, in terms of originating cell (or neighboring hand-in direction) and hand-out cells. After the training stage, the CNN is validated during the new simulations (the historical step has terminated with training): when new hosts make new service request or arrive from a previous neighboring cell, new images are created, and the CNN is validated

(for each cell), that is images are classified. In this way, it is possible to know the class to which any new mobile host belongs, having the possibility to know, in advance, what its next cell will be. In a few words, the main aim behind the entire concept is to encode a certain ordered and limited set of tuples extracted from mh_j mobility in c_i into images which the CNN_i categorizes by returning the sub-set to which the mobility data of node mh_j belongs. To achieve this aim, we propose a mobility-to-image encoding scheme and then train the CNNs to classify the images in the validation datasets. Images can be obtained either from a direct matrix of the data (referred to as Raw Image Representation — RIR, similar to RMD of [13]) or by creating some artifacts for composing the images (referred to as Artifact Image Representation — AIR, similar to MME of [13]).

Let us now illustrate how to encode mobility data into lossless images (we obtain a matrix, which is then stored as an image using a suitable lossless codec). We also introduce a simplified notation for representing coordinates. Each node mh_j from its mobility in c_i (encoded as datasets in ts_i), regardless of the specific mobility sub-datasets, creates a pattern $P_{i,j} = \{(pos^t_{x,i,j}, pos^t_{y,i,j})\}$, where $t = 1, 2, \ldots$ is the discrete time index (i.e., the field $curr_{time}$ in the stored tuples) and $t_l - t_{l-1} = 1/f_s = T$ is the sampling period (the period over which the mobility position is sampled and stored). $P_{i,j}$ can also be denoted $P_{i,j} = \{(pos^1_{x,i,j}, pos^1_{y,i,j}), (pos^2_{x,i,j}, pos^2_{y,i,j}), \ldots, (pos^t_{x,i,j}, pos^t_{y,i,j})\}$. An ordered subset of $P_{i,j}$, $P^{t_0}_{i,j}$, of length w is therefore defined as $P^{t_0,w}_{i,j} \subseteq P_{i,j}$:

$$P^{t_0,w}_{i,j} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_w, y_w)\} \; if \; \exists \; t_0 \; |(x_1, y_1) = (pos^{t_0}_{x,i,j}, pos^{t_0}_{y,i,j}), (x_2, y_2) = (pos^{t_0+1}_{x,i,j}, pos^{t_0+1}_{y,i,j}), (x_w, y_w) = pos^{t_0+w}_{x,i,j}, pos^{t_0+w}_{y,i,j}. \tag{8}$$

This is referred to as a pattern subset. Starting from $P^{t_0,w}_{i,j}$, we can define the related speed $S^{t_0,w}_{i,j}$ of node mh_j in cell c_i in the time range $[t_0, t_0+w]$ as:

$$S^{t_0,w}_{i,j} = \frac{1}{T} \cdot \{pos^{t_0+l+1}_{x,i,j} - pos^{t_0+l}_{x,i,j}, pos^{t_0+l+1}_{y,i,j} - pos^{t_0+l}_{y,i,j}\} \tag{9}$$

where $l = 0, 1, \ldots, w-1$, and $Sx^{t_0,w}_{i,j} = (pos^{t_0+l+1}_{x,i,j} - pos^{t_0+l}_{x,i,j})$, $Sy^{t_0,w}_{i,j} = (pos^{t_0+l+1}_{y,i,j} - pos^{t_0+l}_{y,i,j})$. The length of $S^{t_0,w}_{i,j}$ will be $w-1$. In the same manner, we can also define the acceleration/deceleration $A^{t_0,w}_{i,j}$ of node mh_j in c_i in the time range $[t_0, t_0 + w]$ as:

$$A^{t_0,w}_{i,j} = \frac{1}{T} \cdot \{(Sx^{t_0+q+1,w}_{i,j} - Sx^{t_0+q,w}_{i,j}, Sy^{t_0+q+1,w}_{i,j} - Sy^{t_0+q,w}_{i,j})\}, \tag{10}$$

where $q = 0, 1, \ldots, w-2$, and $Ax^{t_0,w}_{i,j} = (Sx^{t_0+q+1,w}_{i,j} - Sx^{t_0+q,w}_{i,j})$, $Ay^{t_0,w}_{i,j} = (Sy^{t_0+q+1,w}_{i,j} - Sy^{t_0+q,w}_{i,j})$. The length of $a^{t_0}_k$ is $w-2$. Below, we use the shorter notations for $P^{t_0,w}_{i,j}$, $S^{t_0,w}_{i,j}$ and $A^{t_0,w}_{i,j}$. In the case of RIR, the information is converted from a vector to a matrix format. Three matrices are obtained for $P^{t_0,w}_{i,j}$, $S^{t_0,w}_{i,j}$ and $A^{t_0,w}_{i,j}$. Without loss of generality, we assume that for RIR, w is a square value. It is, therefore, easy to define as $w *= \sqrt{w}$. For speed and acceleration vectors, the missing elements (1 and 2, respectively) can be padded during matrix construction (we assume, for example, that the values are replaced with zeros). Matrix dimensions are $w * x\ 2w *$, as $w *$ columns are dedicated to each coordinate.

Algorithm 3 represents a data-structure transformation with a computational complexity of $\theta(w *^2)$. It is applied to position, speed, and acceleration vectors. It accepts as input the vector $vec^{t_0}_j$, which contains position, speed, or acceleration mobility information (specified in the input parameter *type*) related to mobile host mh_j. The algorithm first evaluates the correct matrix dimension $w *$, then executes two nested *For* cycles to fill the rows and columns of the output matrix $M^{t_0}_j$, which was initially set to null, and finally normalizes the matrix in the range $[0, 1]$. Inside the nested loops, the notation $vec^{t_0}_j(c, ind_v)$, where c = 1,2, refers to the x and y components of the $vec^{t_0}_j(c)$ element. Let us now observe how raw data can be effectively encoded by placing it into a matrix structure (AIR). We start with building three separate

Algorithm 3: Raw Image Representation (RIR)

Result: Converts a w-length vector into a matrix format
input: $vec^{t_0}_j$; *type*; w;
start init:
if $type ==' position'$ **then**
  $w *= \sqrt{w}$;
if $type ==' speed'$ **then**
  $w *= \sqrt{w+1}$;
else
  $w *= \sqrt{w+2}$;
end init
create $M^{t_0}_j$=zeros(w*,2·w*); set $ind_v = 1$;
for $h = 1;\ h \le r;\ h++$ **do**
  for $t = 1;\ t \le r;\ t++$ **do**
    $M^{t_0}_j(h, t) = vec^{t_0}_j(1, ind_v)$;
    $M^{t_0}_j(h, t + w *) = vec^{t_0}_j(2, ind_v)$;
    $ind_v = ind_v + 1$;
    if $ind_v > length(vec^{t_0}_j)$ **then**
      break;
normalize($M^{t_0}_j$, 0, 1);
return $M^{t_0}_j$;

matrices from the three vectors. A spreading factor sf is defined, and each matrix is square: setting $w *= (w \cdot sf)$ with dimensions $w^* \cdot w^*$. All the elements of $P^{t_0,w}_{i,j}$, $S^{t_0,w}_{i,j}$ and $A^{t_0,w}_{i,j}$ are initially re-scaled (rs) between 0 and 1 (they may also contain negative values). So:

$$\overline{P}^{t_0}_j = rs(P^{t_0,w}_{i,j}), \quad \overline{S}^{t_0}_j = rs(S^{t_0,w}_{i,j}), \quad \overline{A}^{t_0}_j = rs(A^{t_0,w}_{i,j}). \tag{11}$$

The rs operation consists of dividing all the elements of a vector by the maximum element if it contains no negative values; otherwise, the absolute value of the lowest negative element is added to each value before selecting the maximum and normalizing the elements. The structure of the matrices is then defined as follows. The $w^* \cdot w^*$ matrices $Mp^{t_0}_j$, $Ms^{t_0}_j$ and $Ma^{t_0}_j$ are first created as empty matrices (i.e., each element is equal to zero).

In each matrix, the x component is encoded as a sequence of columns from the top to the bottom, and the y component is encoded as columns from the bottom to the top. The aim is to associate a row or a column of "ones" which is proportional to the content of $\overline{P}^{t_0}_j$, $\overline{S}^{t_0}_j$, $\overline{A}^{t_0}_j$.

This proposed method is only one of the possible methods for filling matrices. We aim to demonstrate the obtainable results and different ways of creating images, with different resulting performance and prediction error/accuracy (optimization of these approaches is beyond the scope of the current work). Algorithm 4 specifies how the three matrices are filled; we illustrate the algorithm for a generic matrix M.

Algorithm 4 accepts as input the normalized vector $vec^{t_0}_j$, which contains position, speed, or acceleration mobility information (specified in the input parameter *type*) related to mobile host mh_j. The length of the subset w and the spreading factor sf are also given as input. The values of sf indicate the number of columns assigned to each component.

AIR first evaluates the previously defined matrix dimension w and creates the empty matrix $M^{t_0}_j$. The algorithm then computes the correct input vector dimension. From Eqs. (9) and (10), we know that the lengths of the speed and acceleration vectors are $w-1$ and $w-2$, respectively. At this point, the algorithm executes the first cycle (h variable) to produce the columns ($index_x$) 1, 1+sf, 1+2·sf, etc. for the x components and the columns ($index_y$), 3, 3+sf, 3+2·sf, etc. for the y components. If the algorithm is applied to a three-dimensional mobility dataset, rows instead of columns can represent the z components, or

Algorithm 4: Artifact to Image Representation (AIR)

Result: Creates a Matrix of artifacts from a vector to encode mobility data

input: $vec_j^{t_0}$; sf; $type$; w;

init: set $w *= (w \cdot sf)$; create $M_j^{t_0}$=zeros(w*,w*); limit=0;

if $type ==' position'$ **then**
  limit=w;
if $type ==' speed'$ **then**
  limit=w-1;
else
  limit=w-2;
for $h = 1;\ h \leq limit;\ h++$ **do**
  $index_x = 1 + (h-1) \cdot sf$; $lim_x = \lfloor vec_j^{t_0}(1,h) \cdot w * \rceil$;
  for $t = 1;\ t \leq lim_x;\ t++$ **do**
    $M_j^{t_0}(t, index_x) = 1$;
  $index_y = 3 + (h-1) \cdot sf$; $lim_y = \lfloor vec_j^{t_0}(2,h) \cdot w * \rceil$;
  for $t = 1;\ t \leq lim_y;\ t++$ **do**
    $M_j^{t_0}(w * -t, index_y) = 1$;
return $M_j^{t_0}$;

alternatively, a larger sf may be helpful. Then, each column is filled with many ones equal to the content of $vec_j^{t_0}(c,h)$ elements.

Recalling that the elements of $vec_k^{t_0}$ are normalized from 0 to 1, the term $\lfloor vec_k^{t_0}(c,h) \cdot r \rceil$ is then bounded to the maximum size of $M_j^{t_0}$. The symbol $\lfloor \rceil$ indicates the rounding operation (the nearest integer is selected). The computational complexity is acceptable because the initial operations have constant complexity (negligible), the main cycle is executed $O(w)$ times, and the internal cycles are executed $O(w*)$ times. The overall computational complexity is thus $O(w \cdot 3 \cdot w*) = O(w \cdot 3 \cdot w \cdot sf)$, which is bounded by $O(w^3)$, and w does not depend on the length of the overall pattern. The RIR and AIR algorithms are both executed for

- Position $P_{i,j}^{t_0,w}$:
  $[Mp_j^{t_0}] = AIR(\overline{P}_j^{t_0}, sf, ' position', w)$; or $[Mp_j^{t_0}] = RIR(\overline{P}_j^{t_0}, ' position', w)$;
- Speed $S_{i,j}^{t_0,w}$:
  $[Ms_j^{t_0}] = AIR(\overline{S}_j^{t_0}, sf, ' speed', w)$; or $[Ms_j^{t_0}] = RIR(\overline{S}_j^{t_0}, ' speed', w)$;
- Acceleration $A_{i,j}^{t_0,w}$:
  $[Ma_j^{t_0}] = AIR(\overline{A}_j^{t_0}, sf, ' accel', w)$; or $[Ma_j^{t_0}] = RIR(\overline{A}_j^{t_0}, ' accel', w)$;

It is important to provide an example of matrices created with the AIR algorithm from real values (RIR is more intuitive since it is a conversion from vectors to matrices only). We use low values for the parameters to produce a concrete and comprehensible example. Let us set $w = 4$, $sf = 2$, $t_0 = 0$ and $P_{i,j}^{0,4} = \{(11.645, 88.734), (16.241, 82.773), (22.750, 76.112), (27.305, 69.552)\}$.

Assuming $T = 1s$ and applying Eqs. (9) and (10), we have $S_{i,j}^{0,3} = \{(-4.596, 5.961), (-6.51, 6.661), (-4.555, 6.56)\}$ m/s, and $A_{i,j}^{0,2} = \{(1.914, -0.7), (-1.854, 0.101)\}\ m/s^2$. Then, applying the rs function, the normalized vectors (Eq. (11)) are

- $\overline{P}_k^0$: $\{(0.131, 1), (0.183, 0.933), (0.256, 0.858), (0.308, 0.784)\}$;
- $\overline{S}_k^0$: $\{(0.14525, 0.94685), (0, 1), (0.14837, 0.99233)\}$;
- $\overline{A}_k^0$: $\{(1, 0.32428), (0, 0.53142)\}$.

Using the input value of sf, we then obtain the number of rows and columns for the matrices, where $r = w \cdot sf = 4 \cdot 2 = 8$, and thus, the three matrices have the dimensions r x r = 8 x 8. The AIR algorithm is applied to each normalized vector, producing the matrices of Fig. 5.

Fig. 5 indicates how the unitary elements add the x components from the top to the bottom (in the columns indexed by AIR as 1, 3, 5, 7) and the y components from the bottom to the top (in the columns indexed by AIR as 2, 4, 6, 8). The next (and final) step is encoding the matrices into an image. To achieve this aim, we multiply each matrix by 2^8-1 (8-bit resolution per layer) and then associate the $Mp_w^{t_0}$ matrix with the RED image channel, the $Ms_w^{t_0}$ matrix with the GREEN image channel and the $Ma_w^{t_0}$ matrix with the BLUE image channel, independently of RIR or AIR. The specific image format is not important, but for our purposes, we used lossless Portable Network Graphics (PNG) encoding [32] to avoid the creation of artifacts (JPEG, for example, is not suitable for our approach). We selected 24-bit RGB PNG; therefore, each element in each matrix is represented by 8 bits (256 levels) of information. A visual magnified representation of the matrices illustrated in Fig. 5 is detailed in Fig. 6.

	1	2	3	4	5	6	7	8
1	1	1	1	0	1	0	1	0
2	0	1	0	1	1	1	1	0
3	0	1	0	1	0	1	0	1
4	0	1	0	1	0	1	0	1
5	0	1	0	1	0	1	0	1
6	0	1	0	1	0	1	0	1
7	0	1	0	1	0	1	0	1
8	0	1	0	1	0	1	0	1

(a) The Mp_4^0 matrix, related to $\overline{P}_{i,j}^{0,2}$.

	1	2	3	4	5	6	7	8
1	1	1	0	1	1	1	0	0
2	0	1	0	1	0	1	0	0
3	0	1	0	1	0	1	0	0
4	0	1	0	1	0	1	0	0
5	0	1	0	1	0	1	0	0
6	0	1	0	1	0	1	0	0
7	0	1	0	1	0	1	0	0
8	0	1	0	1	0	1	0	0

(b) The Ms_4^0 matrix, related to $\overline{S}_{i,j}^{0,2}$

	1	2	3	4	5	6	7	8
1	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0
5	1	0	0	1	0	0	0	0
6	1	1	0	1	0	0	0	0
7	1	1	0	1	0	0	0	0
8	1	1	0	1	0	0	0	0

(c) The Ma_4^0 matrix, related to $\overline{A}_{i,j}^{0,2}$.

Fig. 5. Example of the three matrices as an output of AIR.

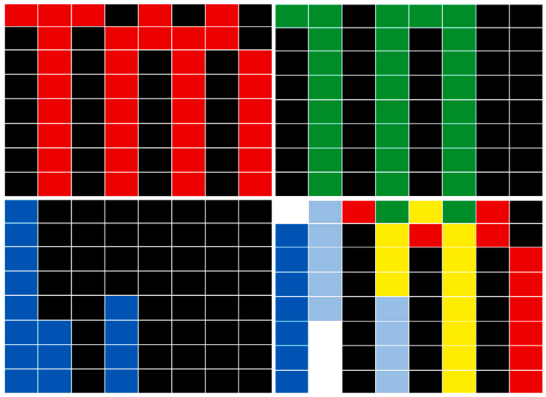


Fig. 6. A representation of $Mp_w^{t_0}$, $Ms_w^{t_0}$, $Ma_w^{t_0}$. Each colored square represents a matrix element (an image pixel): RED channel (position) at the top left, GREEN channel (speed) at the top right, and BLUE channel (acceleration) at the bottom left. The final 8 x 8 image is at the bottom right corner. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Before describing the obtainable results and demonstrating the goodness of our idea, we would like to stress its strengths and weaknesses. STRENGTHS:

- No need to implement a new predictive model: a standard CNN [28] already exists and has demonstrated its strength in image recognition; it can be used to learn local hosts' behavior, and both classify them and predict their future movements;
- The proposed approach is distributed: this means that each considered CNN will be optimized (in terms of training) for its particular coverage area;
- Image data can be easily obtained by periodically sampling mobile host positions inside the coverage area of cell i, with a low-complexity algorithm that is $O(k_0 \cdot k_1)$, where k_1 is the number of images that can be created in function of user's CHT and the sampling period T_s, and k_0 is the number of transmitting mobile hosts in c_i;
- Possibility to extend LeCun's layering structure to improve classification and prediction accuracy (we are showing the performance reachable by the basic structure), which can be considered a future work.

WEAKNESSES:

- The overall algorithm is heavily influenced by the chosen (or available) sampling frequency f_s. It is clear that the time needed to train the CNN depends on f_s, as well as the time necessary to validate (classify and predict) a moving host: as shown in 4, for example, if f_s belongs to the order of tens/hundreds of Hz, then the overall algorithm has a speedy time response e.g. with f_s=100 Hz (T_s=10 ms); the number of images that can be encoded in 1s can vary from 10 to 4, giving a chance to the algorithm to respond in a reasonable time (more-less from 0.1s to 1s). The main issue is the leak of such granularity in real environments where a standard sampling period of 1s is generally considered (f_s=1 Hz). We already outlined the importance of sampling frequency selection in mobile networks [33,34], so choosing random (not theoretically-based) f_s is not always a good idea. Nevertheless, in real scenarios, the training step will be longer than the simulated duration but executed just once. On the other hand, the duration of the validation/prediction step will depend on the time needed to create images: in our simulated case, with w=24 and T_s=10 ms for example, it takes 0.24 s, while in real cases, 24 s could be needed, although good performance can also be reached by reducing the value of samples required.

Fig. 7 shows the time needed to build a complete image, both for AIR (black) and RIR (red). Let us recall that if the final image has a side size dim (dimxdim pixels), then for the AIR algorithm the number of needed samples will be $w = \frac{dim}{sf}$, while it will be dim^2 for the RIR algorithm. In Fig. 7, sf has been set to 2, so it can be seen that the number of samples needed for RIR is unsuitable, except for very high sampling frequencies. The range of w has been considered to be $[10, 24]$ and, as already discussed, it can be seen that for higher values of T_s, the time (i.e. number of samples) needed to build a complete image increases linearly for AIR. As regards the time needed to make a prediction, once the CNN is trained, the desired output can be obtained in less than 20 ms, which is negligible if compared with the time needed to build up an image.

4. Simulation results

This section analyzes the main results of the proposed mobility prediction method and its functional performance in several parameters. For the analysis, we developed a simulator in MATLAB using Python scripts and implemented several *m*-files for the following steps:

- Import a map (GA) from OSM [10] in .osm format;.

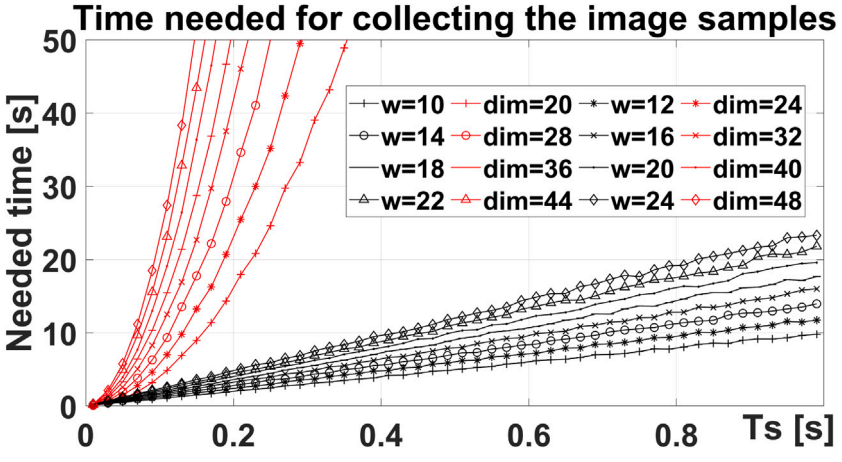
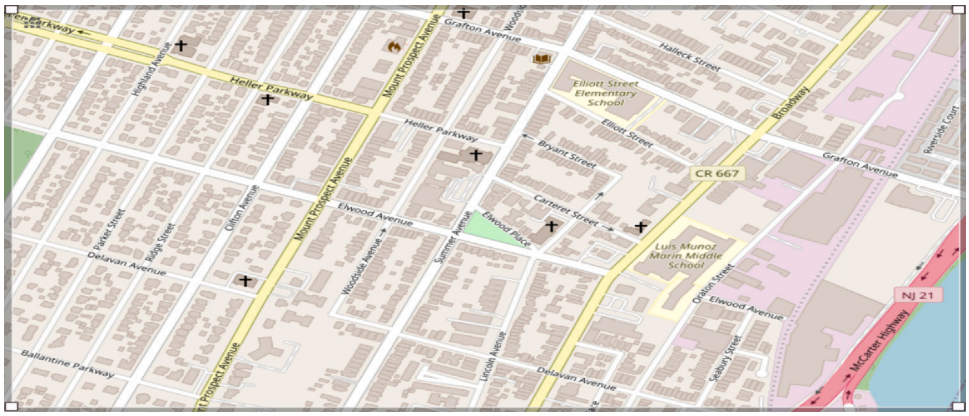


Fig. 7. Average trend of the time needed to create one image, in function of T_s, w (AIR) and dim (RIR).



(a) Residential New York City (USA) GA (GA_1).

(b) Paris (France) GA (GA_2).

(c) Rome (Italy) GA (GA_3).

Fig. 8. Three GAs of 1500 m^2 for simulations.

- Generate real mobility paths on the GA and incorporate a real road structure [11]. We applied the *netconvert.py* and *randomtrips.py* scripts for this step.
- Import mobility coordinate data in SUMO format and convert these data into a MATLAB cell data structure.
- Create a suitable, regular cellular coverage for the GA as a function of a given coverage radius or a Voronoi tessellation as a function of the number of generator points (overlay cellular structure).
- Manage mobility traces (represented as a set of x and y coordinate pairs) for conversion into images and definition of the statuses of each mobile host and thus generate a suitable training dataset. Train the CNNs (Machine Learning Toolbox) with the created dataset for 10 different cells. The CNN layering is the same as the one used in our previous work [13], where it is deeply described in sub-section 3.4.
- Validate each trained CNN for classification/prediction accuracy.
- Predict mobility for users who enter a given cell.

Python scripts permitted us to define several parameters for the simulation and the mobility traces: (a) Mobility profile – a "coach" for a generic car model, (b) Maximum attainable speed of 13.9 m/s, (c) Deviation from the speed limit of 0.1 m/s, (d) Mobility sampling speed (step length) (T_s) of 10 ms, (e) Number of new cars appearing (per second) – 4, (f) Simulation time (per run) of 200 s, (g) Call Holding Time (CHT) – exponentially distributed with μ=180s [31].

As a preliminary example, we generated 10,000 mobility traces for three GAs of equal size $GA_A \approx 1.5$ km^2 (GA_x=1510 m, GA_y=1018 m), illustrated in Fig. 8.

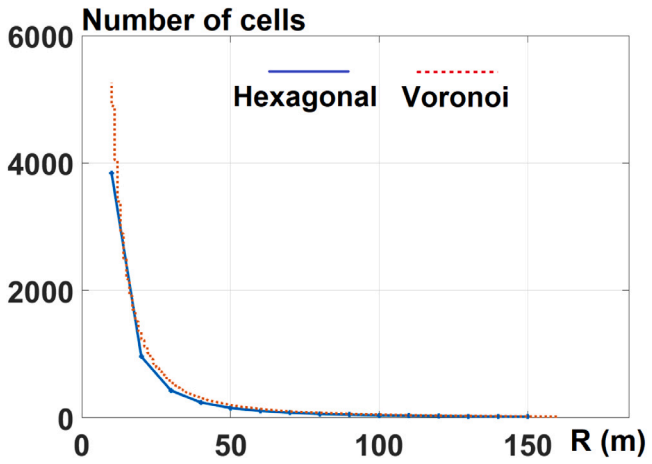


Fig. 9. Total number of required coverage cells as a function of the coverage radius R or $\overline{R}$ for a GA of 1500 m^2.

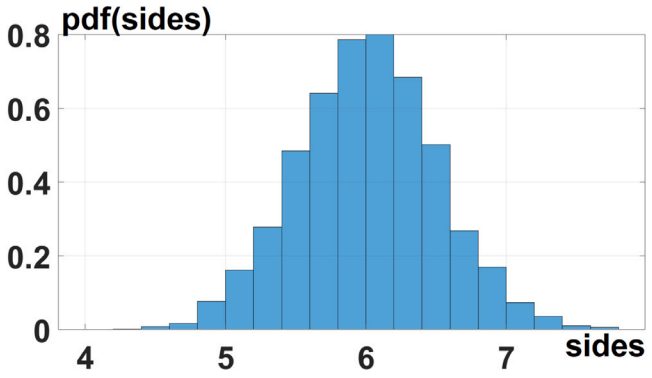


Fig. 10. Probability density distribution of the average number of sides for each cell of a Voronoi tessellation, where GA_A = 1500 m^2, p = 4800.

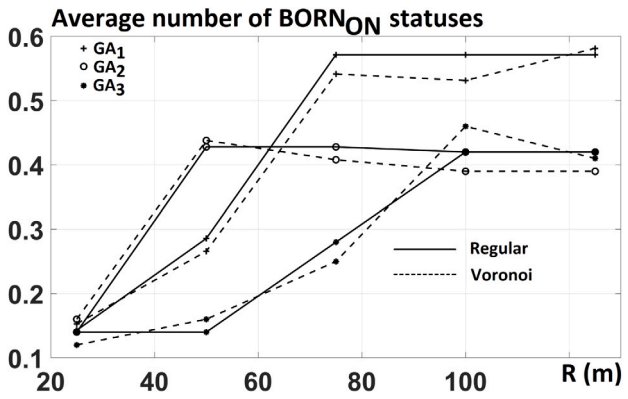


Fig. 11. Average trend of $BORN_{ON}$ status for the maps, coverage shapes, and radii R (25–125 m).

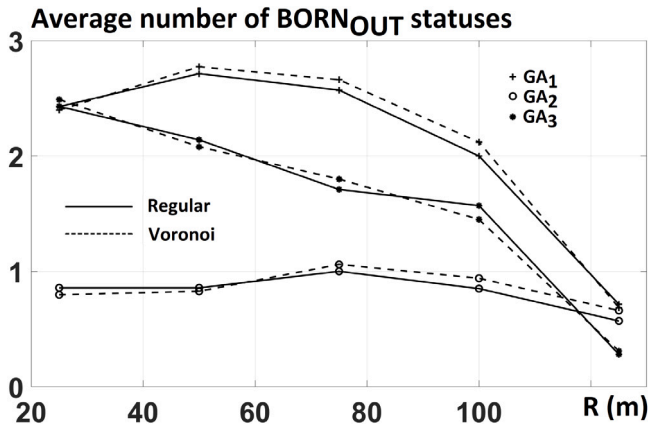


Fig. 12. Average trend of $BORN_{OUT}$ status for the maps, coverage shapes, and radii R (25–125 m).

4.1. Preliminary system analysis

We analyzed both regular (hexagonal) and Voronoi coverage. Fig. 9 plots the number of required cells as a function of the radius for the three 1500 m^2 GAs.

To compare the number of cells for the two types of coverage, we derived an approximation for radius in the Voronoi tessellation:

$$\overline{R} = \sqrt{\frac{1}{\pi \cdot p} \cdot \sum_{i=1}^{p} A(c_i)}. \tag{12}$$

Eq. (12) computes the value of the average radius if the tessellation areas are considered circles of the same coverage.

For the Voronoi tessellation, we initially set the size of GA_p (set of generator points) and subsequently derived each area associated with cell c_i ($A(c_i)$) to be able to apply Eq. (12). For very low coverage radii, the number of required cells is much higher with Voronoi coverage, but after 60/70 m, it becomes comparable to regular coverage. For R = 150 m, the number of cells needed is 10 (it appears to be zero in Fig. 9, but this is a consequence of the scale of the graph in that range). For completeness, we also show the distribution of q_i (number of sides for each Voronoi polygon), an important parameter in the proposed algorithm. Confirming the theory from [35–37], we also obtained a Gaussian distribution for $\mu = 6$ (the distribution is not discrete given that the number of sides is averaged on more than 1000 tessellations for the same area and the same p). The trend and average value are obviously independent of the number of generator points and map extension (see Fig. 10).

At this point, the GAs illustrated in Fig. 8 have the same coverage depicted in Figs. 2 and 3. To build the training set (*historical time*), the simulations commence with the mobility information from approximately 800 mobile hosts on each run (the active connections of mh_j end if the GA is exited or if the $CHT_{i,j}$ expires). Therefore, the tuples for each cell in the GA are generated by mh_j with the coverage of c_i according to Algorithm 1. At the same time, each $c_i \in GA_C$ executes Algorithm 2 to post-process the received tuples and create the correct local STATUSES set, as described in sub- Section 3.3. We recall that the tuples are still formatted as simple text.

Once the *historical time* terminates, we want to determine whether the upper-bound defined in (6) is verified according to the number of STATUSES for each $c_i \in GA_C$.

Fig. 11 indicates a negligible difference between regular (hexagonal) coverage and Voronoi tessellations. At the same time, as the radius increases (larger coverage cells), the number of $BORN_{ON}$ statuses increases. Let us also remember that the values were obtained as the average number of $BORN_{ON}$ statuses for each coverage cell (the value is less than 1 because there are several cells where the connections are handed in or terminated, and depending on the road set, some cells are not able to receive service requests). The increase is due to fewer cells: in a larger area, mobile users begin connecting in the same cell and have a lower probability of affecting other cells.

Fig. 12 shows the trend of the average number of $BORN_{OUT}$ statuses for the entire set of cells. In this case, the trend generally decreases over larger coverage areas: mobile users have a lower probability of exiting their initial coverage area, given the larger radius.

Fig. 13 describes the trend of the average number of hand-over $HAND_{IN-OUT}$ statuses for the entire map. A decreasing trend is

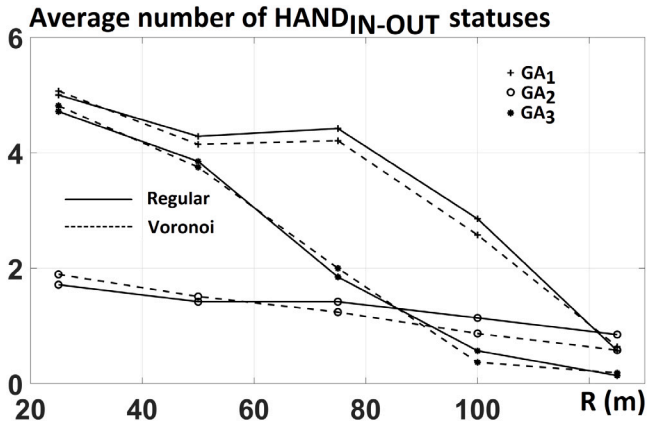


Fig. 13. Average trend of the $HAND_{IN-OUT}$ status for the maps, coverage shapes, and radii R (25–125 m).

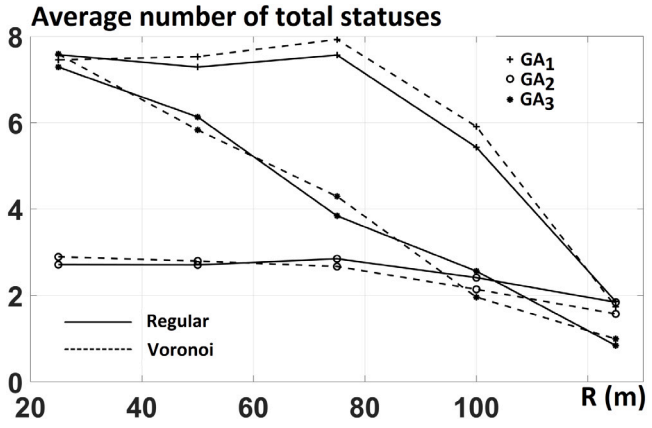


Fig. 14. Average trend of the total number of statuses for the maps, coverage shapes, and radii R (25–125 m).

apparent, indicating that the number of handover events decreases with larger areas.

For simplicity, we did not take into account the trend of $STOP_{IN}$ statuses in each cell since this parameter does not provide any suitable information for prediction purposes (we recall that for this status, mh_j arrives in c_i from any direction IN with an active connection after a hand-over event and ends the connection without handing out c_i).

At this point, it is helpful to know the average number of total statuses as a function of the coverage radius to determine how many classes (sub-datasets) the LeCun-like CNN should manage for training and validation.

Fig. 14 clearly shows the effect on the average number of total statuses (the image classes and sub-datasets for training the CNN). It is clear that this depends on the map topology, the roads R_j traversed by mh_j and the coverage radius, but the theoretical upper bound of Eq. (6), i.e., $(q_i + 1)^2$, is also strongly respected (also neglecting the q_i $STOP_{IN}$ statutes). Assuming that $q_i = 6$ for both hexagonal and Voronoi coverage and ignoring the six $STOP_{IN}$ statuses, we should have a maximum of 43 sub-datasets (whereas the maximum obtained is almost 8 in GA_1, for $R = 75$ m). Eq. (6) then represents the worst case for c_i, considering that the mobile hosts will "born", exit, and traverse it simultaneously.

4.2. Data-to-image conversion and CNN performance

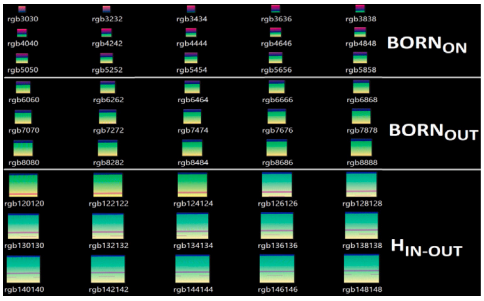
After preliminary execution of the algorithms (i.e., after the historical period), each cell in the system contains (in its memory) the sub-dataset folders for which any event occurred. Therefore, c_i contains each traversed path of mh_j as a sequence of w coordinates groups, i.e., $P_{i,j}^{t_0,w}$, as defined in Eq. (8). Each c_i can execute RIR (Algorithm 3) or AIR a(Algorithm 4) and create a suitable set of images for each sub-dataset folder.

Fig. 15(a) contains examples of images created by the AIR algorithm for three sub-datasets of a cell, where $w = 24$, and Fig. 15(b) contains examples of images made by the RIR algorithm for three sub-datasets of a cell, where $w \in [10, \ldots, 160]$.

Once the training sub-datasets are created for GA_1, GA_2, and GA_3, LeCun's CNN is trained, and its classification/prediction accuracy is validated. Fig. 16 provides an important observation that the RIR conversion algorithm is unsuitable for our purposes. We recall that RIR is based on a direct conversion of coordinates from a vector representation to a matrix (and then to an image) representation without creating any artifacts. The validation process demonstrates some unacceptable results in terms of accuracy (below 45%) for different values of w. The graph in Fig. 16 is limited to a coverage radius of 75 meters and a regular coverage scenario for space reasons. At other radii, the results obtained for RIR validation always fall below 50%, mainly because without image artifacts, each CNN_i belonging to c_i cannot extract sufficient features. We had to consider the range [8, 56] for w, because at higher values, RIR needs more than 56 x 56 = 3156 mobility samples for each image, which is not always possible if a dense training dataset is required.

To illustrate, Fig. 17 plots the image size (in pixels) for RIR and AIR as a function of the number of mobility samples w. RIR requires a massive number of samples to create bigger images (the values for RIR are not comparable to those of AIR at the same w, i.e., from 5 to 60). This means that RIR needs 100 samples to create an image of 10 x 10 pixels, whereas AIR needs only 10 samples at the minimum $sf = 2$.

(a) AIR images for three training subsets: $BORN_{ON}$, $BORN_{OUT}$ and $HAND_{IN-OUT}$, for $w = 24$ and $sf = 4$.



(b) Example of RIR images obtained for three training subsets: $BORN_{ON}$, $BORN_{OUT}$ and $HAND_{IN-OUT}$, for w varying from 10 to 160.

Fig. 15. Examples of encoded mobility with AIR and RIR (the number below each image is the progressive name for each item).

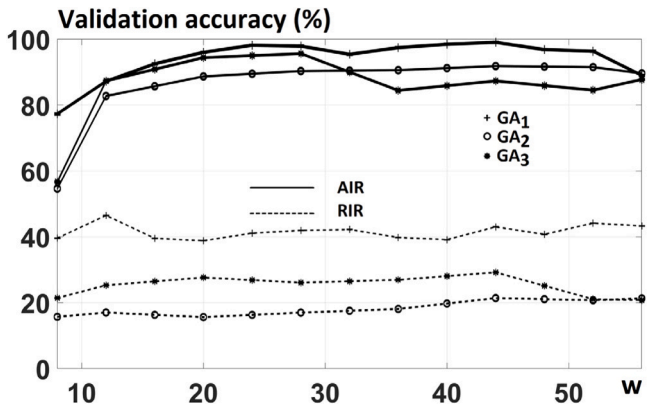


Fig. 16. Validation accuracy for the AIR and RIR algorithms as a function of the subset observation size $w \in [8..56]$ (regular coverage at $R = 75$ m).

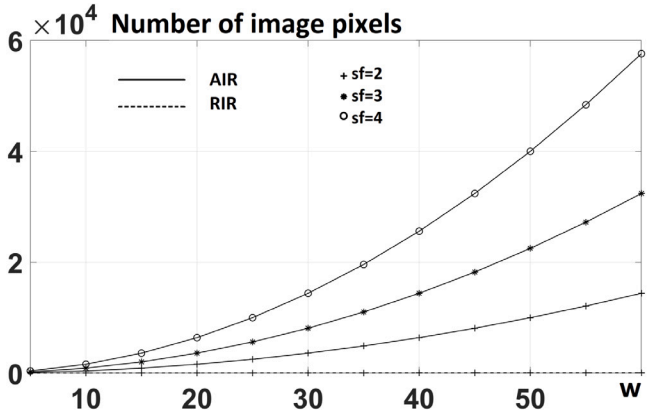


Fig. 17. Number of image pixels for RIR and AIR as a function of the number of samples (observation subset size) $w \in [5..60]$.

Compared to RIR, AIR also demonstrates greater robustness with a very low number of samples.

Henceforth, given its several advantages (especially validation accuracy), we show only the results obtained with AIR. First, the obtained image samples were pruned to train the CNN adequately with a minimum number of samples in each class (status). Any status with less than 50 related images was excluded as invalid. For reasons of limited space, only the results for $R = 75$ m and regular coverage scenario are shown, and also because higher values of R result in fewer hand-over events (hence, at lower numbers of statuses, as shown in Fig. 14) and lower values result in each status having fewer images and requiring much longer simulation times. The plotted values were averaged over the number of coverage cells (110, the same for each GA).

Fig. 18 shows the trend of the average number of valid statuses and that it does not depend heavily on w or sf, only on the considered GA. The trend is almost constant (higher trends for GA_1 and GA_2 at low w and average sf relate only to the different events generated during different simulations).

Fig. 19 describes the trend of the average number of elements (images) per training class (status). A lower W produces tremendous values (fewer coordinates are required to encode an image), and the results for the three GAs overlap, confirming that the proposed method is independent of the geographic area. The minimum value is 50, and the average maximum is approximately 664. The spreading factor does not affect the trend since it acts only on the effective dimensions of each image. To train the CNN of each cell, 70% of the images of each class were used, and the remaining 30% were used for validation.

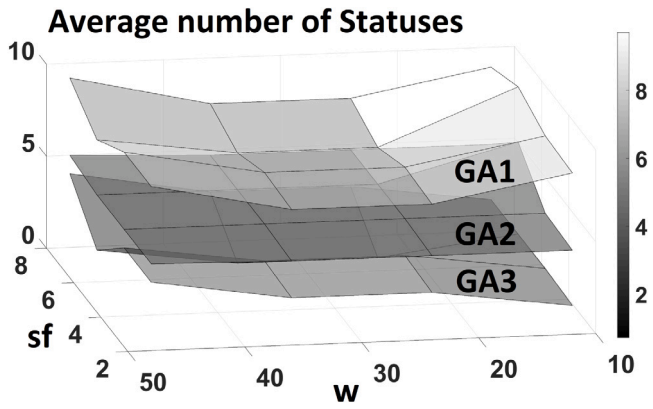


Fig. 18. Average number of statuses containing a significant number of samples for different values of w and sf.

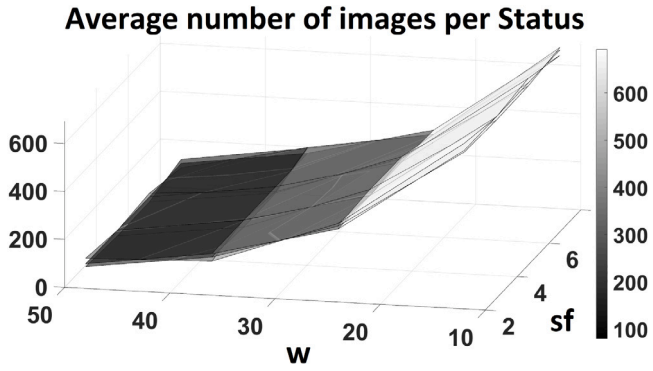


Fig. 19. Trend of the average number of elements for each training class (status) for different values of w and sf.

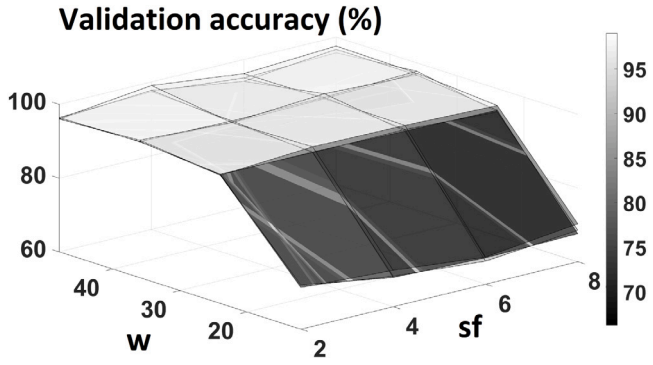


Fig. 20. Resulting trend of the average validation accuracy for different values of w and sf.

Fig. 20 shows the proposed method's validation (prediction) accuracy. In this case, the results are also independent of the geographic area. Generally, for a higher sf, we observe a slightly decreasing trend (because the created features become sparser), and at $sf = 4$, the system achieves the best average accuracy (97.34%). Regarding the effect of w on accuracy, each CNN requires a minimum image size to perform satisfactorily; w should be set to at least 24 (image size 48 x 48 pixels at $sf = 2$, 96 x 96 pixels at $sf = 4$, 144 x 144 at $sf = 6$ and 208 x 208 at $sf = 8$). From the simulated values, we conclude that each CNN can perform with very high accuracy at $w = 24$ and $sf = 4$. No proposed works in literature achieve a comparable result.

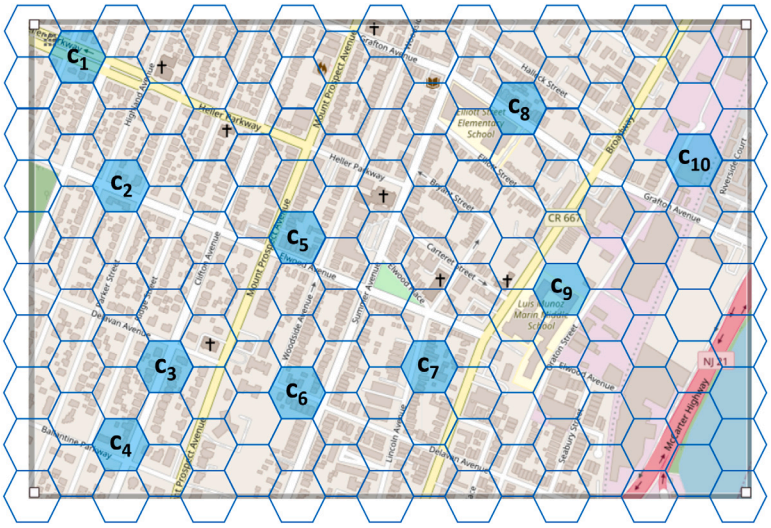


Fig. 21. An example of the considered simulated maps (New York) and the coverage cells (R = 50 m) used for validating our approach. The indexed and colored cells are randomly chosen for each simulation run. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.3. Mobility prediction through the trained CNN deployment

In this sub-section, the results related to our proposal in terms of prediction accuracy and CDP are presented as a case study about the average results of the next-cell prediction CNN approach, described in the previous sections. In particular, one CNN has been assigned to each considered coverage cell, and the training procedure has been executed. An example of the considered simulation scenarios is depicted in Fig. 21, where the New York map has been covered by a regular-shape tessellation with a coverage range of 50 m. Ten cells are randomly chosen for each run (a total 100). For each map, for different coverage ranges (from 25 m to 200 m with a step of 25 m), then their CNNs are trained and, in the end, the Distributed Prediction with Bandwidth Management Algorithm (DPBMA) framework of [30] has been used, without considering the Markovian model, as well as the concept of roads compression, but just associating a CNN to each cell. We indicate the new framework as Image-based DPBMA (I-DPBMA). So, each trained CNN has been used as the core for predicting users' movements and reserving the appropriate bandwidth before users' arrival. To give more emphasis to the obtained results, we also considered what happens when a real coverage map is used to train our proposal, as made in several works, such as [38]. In particular, the UMTS Voronoi tessellation map of subsection 4.3 (Figures 24 and 29) of [30] has been considered, as well as the subset of 10 cells: $C_{real} = \{c_1, c_3, c_4, c_6, c_{10}, c_{11}, c_{12}, c_{16}, c_{17}, c_{19}\}$. On such a dataset, we did not have the chance to change either the coverage radius (mean value $\overline{R}$ of 1.493 km, evaluated as in Eq. (12)), or the mobility model, that is a Kraub mobility model [39] (with maximum acceleration equal to 1.4 m/s^2, maximum deceleration equal to 2 m/s^2, σ= 0.5, τ =0.2 s and T_s=1s). So, black curves refer to the I-DPBMA applied to the three different maps with a small-cells scenario, as depicted in Fig. 21, while the colored segments refer to mobility data used in subsection 4.3 of [30], for the DPBMA (red) and I-DPBMA (green). In particular, Fig. 22 shows the trend of the CDP, averaged on the ten cells and the number of simulations. It can be seen that, for our new-proposed AIR encoding (black curves), CDP is almost negligible because it maintains below the value of 0.05. It must be noticed that a mobile connection is dropped (CDP==1) when the mobile host does not find the available resources in the new cell due to a prediction error or to the bandwidth unavailability in the next cell when the service request is made [30,40]. At the same time, although a next-cell prediction is wrong, a mobile host may find the necessary resources available. On average, DPBMA shows higher CDP, which is outperformed by I-DPBMA. Fig. 23 shows

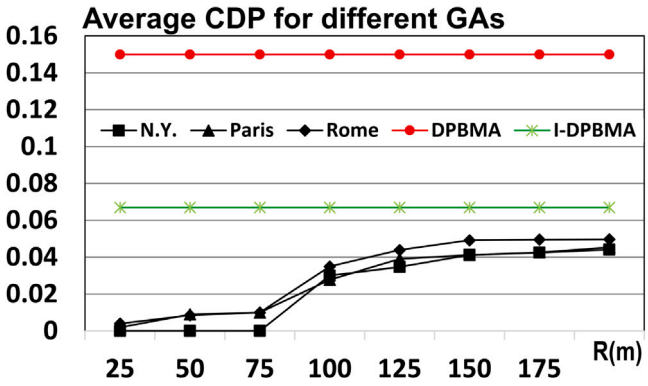


Fig. 22. The average trend of the CDP evaluated at the first hand-over event.

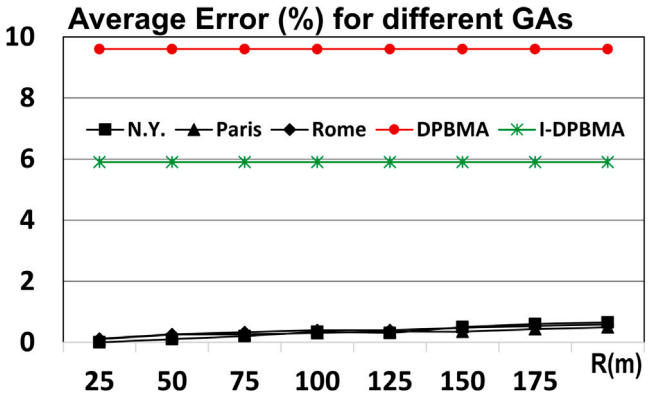


Fig. 23. The average trend of the next-cell prediction error (%).

the overall average prediction error: this metric is more precise than the previous one because it refers to the percentage of wrong predictions (in the average). It is evident how the obtained error maintains below 1%, which means that the error can be completely neglected for a 5G/6G cellular environment, where small-cell deployment will be huge. If real coverage is considered (DPBMA and I-DPBMA), the error increases, although it maintains below 10% (DPBMA), and is outperformed by I-DPBMA (below 6%).

5. Conclusions and future work

In the current work, we proposed a new approach for mobility prediction (especially next-cell prediction) based on encoding data as images. We avoided introducing any new neural network layering. Instead, we exploited the strength of CNNs in image classification by converting mobility samples (coordinates) into images and adding simple features derived from mobility inputs. We demonstrated that the complexity of the proposed scheme is polynomial and that image encoding can be easily performed by each covering cell. We performed several simulations with suitable CNN training and achieved an accuracy of over 97%.

In future work, we plan to optimize the encoding algorithm by investigating methods of creating image features and associating the correct entropy level required by CNNs to learn the image content more efficiently.

CRediT authorship contribution statement

Peppino Fazio: Writing – review & editing, Writing – original draft, Supervision, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Miralem Mehic:** Writing – review & editing, Writing – original draft, Supervision, Software, Formal analysis, Conceptualization. **Miroslav Voznak:** Writing – review & editing, Validation, Supervision, Resources, Project administration, Funding acquisition, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by the European Union within the REFRESH project – Research Excellence For Region Sustainability and High-tech Industries ID No. CZ.10.03.01/00/22003/0000048 of the European Just Transition Fund. This work was also supported by the Ministry of Science, Higher Education and Youth of Canton Sarajevo, Bosnia and Herzegovina under Grant No. 27-02-35-33081-10/24.

References

[1] G. Liu, Y. Huang, N. Li, J. Dong, J. Jin, Q. Wang, N. Li, Vision, requirements and network architecture of 6G mobile network beyond 2030, China Commun. 17 (9) (2020) 92–104, http://dx.doi.org/10.23919/JCC.2020.09.008.

[2] P. Fazio, et al., Prediction and QoS enhancement in new generation cellular networks with mobile hosts: A survey on different protocols and conventional/unconventional approaches, IEEE Commun. Surv. Tutor. 19 (3) (2017) 1822–1841, http://dx.doi.org/10.1109/COMST.2017.2684778.

[3] T. Martin, S. Areibi, G. Grewal, Effective machine-learning models for predicting routability during FPGA placement, in: 2021 ACM/IEEE 3rd Workshop on Machine Learning for CAD, MLCAD, 2021, pp. 1–6, http://dx.doi.org/10.1109/MLCAD52597.2021.9531243.

[4] A. Singh, S. Sharma, A. Gumaste, Using deep reinforcement learning for routing in IP networks, in: 2021 International Conference on Computer Communications and Networks, ICCCN, 2021, pp. 1–9, http://dx.doi.org/10.1109/ICCCN52240.2021.9522197.

[5] R. Xu, W. Li, K. Li, X. Zhou, H. Qi, DarkTE: Towards dark traffic engineering in data center networks with ensemble learning, in: 2021 IEEE/ACM 29th International Symposium on Quality of Service, IWQOS, 2021, pp. 1–10, http://dx.doi.org/10.1109/IWQOS52092.2021.9521298.

[6] S. Khan, H. Rahmani, S.A.A. Shah, M. Bennamoun, G. Medioni, S. Dickinson, A guide to convolutional neural networks for computer vision, in: Morgan & Claypool Series, 2018, http://dx.doi.org/10.1007/978-3-031-01821-3.

[7] C. Luo, X. Li, L. Wang, J. He, D. Li, J. Zhou, How does the data set affect CNN-based image classification performance? in: 2018 5th International Conference on Systems and Informatics, ICSAI, 2018, pp. 361–366, http://dx.doi.org/10.1109/ICSAI.2018.8599448.

[8] H. Zhang, L. Dai, Mobility prediction: A survey on state-of-the-art schemes and future applications, IEEE Access 7 (2019) 802–822, http://dx.doi.org/10.1109/ACCESS.2018.2885821.

[9] S.J. Frank, A.M. Frank, Salient slices: Improved neural network training and performance with image entropy, Neural Comput. 10 (1162) (2020) 1222–1237, http://dx.doi.org/10.48550/arXiv.1907.12436.

[10] Open Street Map (OSM).

[11] P.A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, E. Wiessner, Microscopic traffic simulation using SUMO, in: 2018 21st International Conference on Intelligent Transportation Systems, ITSC, 2018, pp. 2575–2582, http://dx.doi.org/10.1109/ITSC.2018.8569938.

[12] V.V. Chetlur, H.S. Dhillon, On the load distribution of vehicular users modeled by a Poisson line cox process, IEEE Wirel. Commun. Lett. 9 (12) (2020) 2121–2125, http://dx.doi.org/10.1109/LWC.2020.3014585.

[13] P. Fazio, et al., A novel urban mobility classification approach based on convolutional neural networks and mobility-to-image encoding, J. King Saud Univ. - Comput. Inf. Sci. 35 (6) (2023) 1822–1841.

[14] M.F. Mosleh, M.J. Zaiter, A.H. Hashim, Position estimation using trilateration based on ToA/RSS and AoA measurement, J. Phys.: Conf. Ser. 1773 (2021) http://dx.doi.org/10.1088/1742-6596/1773/1/012002.

[15] L. Huang, L. Lu, W. Hua, A survey on next-cell prediction in cellular networks: Schemes and applications, IEEE Access 8 (2020) 201468–201485, http://dx.doi.org/10.1109/ACCESS.2020.3036070.

[16] H. Wang, Y. Li, D. Jin, Z. Han, Attentional Markov model for human mobility prediction, IEEE J. Sel. Areas Commun. 39 (7) (2021) 2213–2225, http://dx.doi.org/10.1109/JSAC.2021.3078499.

[17] X. Hu, W. Liu, H. Huo, An intelligent network traffic prediction method based on butterworth filter and CNN–LSTM, Comput. Netw. 240 (2024) 110172, http://dx.doi.org/10.1016/j.comnet.2024.110172.

[18] Z. Jin, J. Qian, Z. Kong, C. Pan, A mobility aware network traffic prediction model based on dynamic graph attention spatio-temporal network, Comput. Netw. 235 (2023) 109981, http://dx.doi.org/10.1016/j.comnet.2023.109981.

[19] G. Yu, Y. He, J. Wu, Z. Chen, J. Pan, Mobility-aware proactive edge caching for large files in the internet of vehicles, IEEE Internet Things J. 10 (13) (2023) 11293–11305, http://dx.doi.org/10.1109/JIOT.2023.3240423.

[20] K. Cao, Y. Liu, L. Duan, S. Xu, H. Jung, Research on regional traffic flow prediction based on MGCN-WOALSTM, IEEE Access 11 (2023) 126436–126446, http://dx.doi.org/10.1109/ACCESS.2023.3330909.

[21] H. Gao, R. Jiang, Z. Dong, J. Deng, Y. Ma, X. Song, Spatial-temporal-decoupled masked pre-training for spatiotemporal forecasting, 2024, http://dx.doi.org/10.48550/arXiv.2312.00516, arXiv:2312.00516.

[22] L. Zhai, Y. Wang, S. Cui, Y. Zhou, A comprehensive review of deep learning-based real-world image restoration, IEEE Access 11 (2023) 21049–21067, http://dx.doi.org/10.1109/ACCESS.2023.3250616.

[23] Y. Pei, Y. Huang, Q. Zou, X. Zhang, S. Wang, Effects of image degradation and degradation removal to CNN-based image classification, IEEE Trans. Pattern Anal. Mach. Intell. 43 (4) (2021) 1239–1253, http://dx.doi.org/10.1109/TPAMI.2019.2950923.

[24] Y. Xu, W. Liu, K.F. Kelly, Compressed domain image classification using a dynamic-rate neural network, IEEE Access 8 (2020) 217711–217722, http://dx.doi.org/10.1109/ACCESS.2020.3041807.

[25] S.A. Nene, S.K. Nayar, H. Mur, Columbia Object Image Library COIL-100, https://www.kaggle.com/jessicali9530/coil100.

[26] V. Tiwari, C. Pandey, A. Dwivedi, V. Yadav, Image classification using deep neural network, in: 2020 2nd International Conference on Advances in Computing, Communication Control and Networking, ICACCCN, 2020, pp. 730–733, http://dx.doi.org/10.1109/ICACCCN51052.2020.9362804.

[27] J. Zhang, C. Chen, B. Li, L. Lyu, S. Wu, S. Ding, C. Shen, C. Wu, DENSE: Data-free one-shot federated learning, in: Advances in Neural Information Processing Systems 35 (NeurIPS 2022), 2022, pp. 21414–21428, http://dx.doi.org/10.48550/arXiv.2112.12371.

[28] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324, http://dx.doi.org/10.1109/5.726791.

[29] S. Won, S.W. Choi, Three decades of 3GPP target cell search through 3G, 4G, and 5G, IEEE Access 8 (2020) 116914–116960, http://dx.doi.org/10.1109/ACCESS.2020.3003012.

[30] P. Fazio, M. Tropea, F.D. Rango, M. Voznak, Pattern prediction and passive bandwidth management for hand-over optimization in QoS cellular networks with vehicular mobility, IEEE Trans. Mob. Comput. 15 (11) (2016) 2809–2824, http://dx.doi.org/10.1109/TMC.2016.2516996.

[31] B.-H. Ku, Y. Ren, J.-F. Weng, J.-C. Chen, W.-T. Chen, Modeling and analysis of channel holding time and handoff rate for packet sessions in all-IP cellular networks, IEEE Trans. Veh. Technol. 66 (4) (2017) 3331–3344, http://dx.doi.org/10.1109/TVT.2016.2588324.

[32] The Portable Network Graphics specification and W3C.
[33] P. Fazio, M. Mehic, M. Voznak, Effects of sampling frequency on node mobility prediction in dynamic networks: A spectral view, Digit. Commun. Netw. 9 (4) (2023) 1009–1022, http://dx.doi.org/10.1016/j.dcan.2022.05.008.
[34] P. Fazio, M. Mehic, F. De Rango, M. Tropea, M. Voznak, Optimization of mobility sampling in dynamic networks using predictive wavelet analysis, Pervasive Mob. Comput. 98 (2024) 101887, http://dx.doi.org/10.1016/j.pmcj.2024.101887.
[35] A.-E. Baert, D. Seme, Voronoi mobile cellular networks: topological properties, in: Third International Symposium on Parallel and Distributed Computing/Third International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks, 2004, pp. 29–35, http://dx.doi.org/10.1109/ISPDC.2004.58.
[36] R. Ullah, N. Fisal, H. Safdar, W. Maqbool, Z. Khalid, A. Khan, Voronoi cell geometry based dynamic fractional frequency reuse for OFDMA cellular networks, in: 2013 IEEE International Conference on Signal and Image Processing Applications, 2013, pp. 435–440, http://dx.doi.org/10.1109/ICSIPA.2013.6708046.
[37] D.F. Watson, The number of edges per face in a large aggregate of space-filling, random-sized, randomly arranged polyhedra, Math. Geol. 7 (1007) (1975) 349–354, http://dx.doi.org/10.1007/BF02081706.
[38] M. Yan, S. Li, C.A. Chan, Y. Shen, Y. Yu, Mobility prediction using a weighted Markov model based on mobile user classification, Sensors 21 (5) (2021) http://dx.doi.org/10.3390/s21051740.
[39] F.J. Martinez, J.-C. Cano, C.T. Calafate, P. Manzoni, CityMob: A mobility model pattern generator for VANETs, in: ICC Workshops - 2008 IEEE International Conference on Communications Workshops, 2008, pp. 370–374, http://dx.doi.org/10.1109/ICCW.2008.76.
[40] F. De Rango, P. Fazio, A stochastic approach for resource prediction error and bandwidth wastage evaluation in advanced dynamic reservation strategies, IEEE Trans. Mob. Comput. 22 (9) (2023) 4986–5000, http://dx.doi.org/10.1109/TMC.2022.3176046.

Peppino Fazio was born 1977 in Italy, he received the PhD. degree in Electronics and Communications Engineering, University of Calabria (UNICAL - Italy) in 2008 and completed his habilitation as Associate Professor in 2017, after being an Assistant Professor at DIMES Dept. (UNICAL) until 2016. Now he is Assistant Professor at the Ca' Foscari University of venice, in the Department of Molecular Sciences and Nanosystems. He is co-author of more than 120 papers (52 in International Journals), all indexed in Scopus and/or WoS. His reputation in the open community network Research Gate, measured by RGScore (885.2), is higher than 92% of ResearchGate members. His research interests include mobile communication networks, QoS architectures and interworking, wireless and wired networks, mobility modeling for WLAN environments, mobility analysis for prediction purposes, routing, vehicular networking, MANET and VANET. He is peer reviewer and TPC member of different international conferences, as well as for many international journals, such as IEEE TVT, COMMLETT, VTM, SPRINGER TELS, MONET, ELSEVIER VEHCOMM, COMNET, and many others.

Miralem Mehic (Member, IEEE) received his Ph.D. degree in telecommunications from the VSB-Technical University of Ostrava (Czechia). Also, he studied at the AGH University of Science and Technology, Krakow, Poland, Alpen-Adria-Universitat Klagenfurt, Austria and Austrian Institute of Technology (AIT) in Department of Digital Safety & Security Business Units - Optical Quantum Technology, Vienna & Klagenfurt, Austria. He is the author of the unique QKD network simulator QKDNETSIM (www.qkdnetsim.info). His field of research is related to the cyber-security, quantum cryptography, quality of service and network management. He is the national principal investigator of EU H2020 OPENQKD and the NATO SPS QUANTUM5 projects.

Miroslav Voznak (Senior Member, IEEE) received his Ph.D. degree in telecommunications from the Faculty of Electrical Engineering and Computer Science, VSB - Technical University of Ostrava in 2002 and completed his habilitation in 2009. He was appointed Full Professor in 2017 in Electronics and Communications technologies. His research interests are generally focused on information and communications technologies, especially quality of service and experience, network security, wireless networks, and big data analytics. He has authored and co-authored about two hundred articles indexed in SCI/SCIE journals. He is included in the list of the World's Top 2% of Scientists in the field of I&CT by Sanford University in 2020, 2021, and 2022. He participated in six research projects funded by European Commission and is currently the PI of the NATO SPS G5894 QUANTUM5 project.