Indoor-Outdoor Detection using Time Series Classification and User Behavioral Cognition

Sid Ali Hamideche, Marie-Line Alberi Morel, Kamal Singh, Cesar Viho

Indoor-Outdoor Detection using Time Series Classification and User Behavioral Cognition

Sid Ali Hamideche
Nokia Paris Saclay
Nozay, France
sid.hamideche@nokia.com

Marie Line Alberi Morel
Nokia Paris Saclay
Nozay, France
marie_line.alberi-morel@nokia.com

Kamal Singh
Laboratoire Hubert Curien
Saint-Etienne, France
kamal.singh@univ-st-etienne.fr

Cesar Viho
IRISA-UR1
Rennes, France
cesar.viho@irisa.fr

***Abstract*—During the last decades, different studies highlighted the benefits of acquiring channel state cognition based on the environmental context of mobile users or devices. Thanks to this cognition, cellular networks can optimize themselves and personalize the delivered services and in turn, offer a better quality of experience to users. This benefit for mobile networks will come only if the environments are detected with high accuracy, short delays and minimal implementation cost. However, accurate environment detection is challenging for mobile networks as real-life situations are numerous, complex and dynamic. In this paper, we investigate the detection of mobile users' environment, in real-life situations, using machine learning classification methods on time series data. To attain the highest accuracy, while using limited length of time series, we propose using a heuristic method to account for the typical user behavior when he or she changes environment. For this, a new module, called User Behavioral Optimizer, is investigated and combined with time series models. It detects erroneous user behaviour predictions output by the machine learning models and then corrects some of them. Experiments are done using real radio data, that has been massively gathered from diverse real situations of mobile users. Experiments show that machine learning on time series data using our behavioral optimizer and heuristic allows to detect indoor/outdoor with $F1-score$, up to around 94.8%.**

***Index Terms*—Indoor/Outdoor Detection, Time Series Classification, LSTM, User Behavioral Optimizer, Machine learning**

I. Introduction

5G-advanced is the step before 6G which will open the door for many new services such as *network as a sensor*. With the later, the mobile network will be able to sense the environment. Thus, the network will become a center of situational information as well as collection and processing of signals and data. One of the idea towards realising the above vision is to design mobile networks that know the context in which the mobile devices are used. Indeed, adding context awareness makes networks more aware of the situations in which mobile users prefer to consume their services and applications. By detecting users' consumption habits, the network infrastructure will be able to efficiently take appropriate decisions in face of variable network conditions and users' habits.

Among the contextual information, (e.g. time, spatial location or social situation) the environmental situation constitutes relevant information about users' activities. For example, users may only listen to music or audio when they are walking outside, while they may use more variety of applications when they are at home, where they may play games or watch movies etc. Some works show that it is possible to beneficially exploit the environmental context. For example, it can be used to enhance network operations, in terms of better QoE for video applications [1], improved handover process [2], slice selection to switch from a slice with more flexible resources to a resilient one [3], accurate user localisation detection [4], etc. Therefore to ensure that the network functions benefit from the cognition about environment, there is a need to design a new function, which can accurately and automatically detect, in real-time, where a mobile user likes to consume mobile services. For this, the detection function is required to be placed close to network functions using environment information to have a win-win situation for Mobile Network Operators (MNOs). Furthermore, ensuring a full automatic process, the functionality should work without requiring constant user interaction. Hence, placing such function inside *infrastructure of operator network* would also have the following advantages: not overloading User Equipment's (UEs) with additional computations and avoiding costly additional signalling for UE reports of environment information or data to network. UEs often have limited data uplink, computing power and energy. So, the challenge is to develop models for mobile networks which are able to quickly detect real-life environments (as well as changes in environment) of connected users with the highest detection accuracy. Indeed longer execution times lead to delayed detection and can be prejudicial for wireless networks that would like to use the information about the user environment. Nonetheless, this is a hard task because there exist various as well as complex real-life environmental situations.

In this paper, we focus on real-life *Users' Environment Detection (UED)*, namely whether the user is indoor or outdoor, using Deep Learning techniques (DL) which learn on mutltivariate time series. Time series classification techniques are popular for their ability to extract additional information from sequences of raw data [5]. In our case, the time information is related to time variations of the input features. Our hypothesis is that, as stated in literature, capturing signal change pattern in time series would help detection algorithms to increase accuracy by better distinguishing ambiguous points. They sometimes have similar instantaneous values even if they correspond to different environments (Indoor vs. Outdoor). We target multivariate time series-based classification models for networks that use relevant environmental features, preferably, extracted from already available data within the network. Recently, some works in [6], [7] showed that UED can be

ISBN 978-3-903176-52-2

achieved accurately using Feed forward Neural Networks (FNN) classification models with specific 3GPP radio signals that are already available at Radio Access Network (RAN) side. We are interested in investigating environment detection using the same 3GPP radio features. But contrary to these previous works, we consider Long-Short-Term-Memory (LSTM) which is a variant of Recursive Neural Network (RNN) models which can process time-ordered sequences of data values obtained successively [8]. Thanks to its complex internal structure, LSTMs can efficiently learn short and long terms sequence correlations. This allows them to model complex multivariate sequences. They outperform non-ML and other deep neural networks approaches in various domains [9]. However using time series data also has drawbacks in terms of training time, delayed detection of environment change, storage space and complexity while classification. So, to ensure high accuracy UED with short time series data model, we propose a new approach which leverages the domain knowledge on how a user typically behaves when he or she switches their environment. For example, a user can not quickly change their environments several times. Thus, using such kind of domain knowledge, we propose a module called *User Behavioral Optimizer (UBO)*. It aims to correct environment detection errors by tracking behavioral anomalies, that diverge from typical behavior. Thus, we establish an anomaly list after considering how a user typically moves from one environment to another. These behavioral anomalies serve to prohibit ML model to change the predicted class too fast and in turn, help to perform accurate UED. We investigate the application of UBO in two ways. First, it is introduced in the training phase of UED model by helping weights and bias computation. Second, it is applied after UED model output.

Our contributions are as follows. We investigate user environment detection using supervised, multivariate, time series classification technique. We study LSTM and also integrate the heuristics corresponding to user behavior in the training phase and at model output stage. Two evaluation methods, *Forward-chaining cross-validation* and *Time Block-based validation*, adapted to time series, are used to prevent data leakage and assess model performance robustly. Bench-marking is conduced to compare the use of LSTMs for a real-time UED with ML and other DL techniques. UBO shows positive impact in terms of reducing time series duration required for high accuracy. As a consequence, such time series models will be preferred for use in real-time context for wireless networks.

This paper is organized as follows. Sec. II presents related work. Sec. III discusses data, motivations and the proposed method. Sec. V presents performance analysis and discussion. Finally, Sec. VI provides conclusions and perspectives.

II. Related Work

In recent years, many works have studied the issue of mobile user environment awareness. Most of them focus on binary classification with two classes (Indoor/Outdoor). Remaining existing works consider up to eight classes: Indoor/Semi-Outdoor/Outdoor [10] or eight types of Indoor/Outdoor environments in [7].

The Indoor/Outdoor detection (IOD) problem in mobile terminal has been predominantly studied on the mobile terminal side. The methods investigated mainly exploit the data produced by the sensors equipped on cell phones, such as GPS, cell signal strength, light intensity as well as magnetic sensor.

On the other hand, there are very few works which study the IOD problem from network side. In this case, IOD model uses features which are available in the network infrastructure or RAN. These works investigate methods that exploit instantaneous measurements and do not consider sequence of temporal data or time series. In [1] the authors use $RSRP$ (Reference Signal Received Power) and $RSRQ$ (Reference Signal Received Quality) signals for IOD and compare Support Vector Machine (SVM), logistic regression and random forest. SVM was found to perform best. In [11], a semi-supervised learning algorithm using a logistic regression model is proposed to estimate the probability that a particular connection was generated indoor. The detection is based on LTE measurements from radio connection traces. [12] studied quantum machine learning approaches for IOD using the 3GPP signals such as received power level and Signal-to-Interference Ratio (SIR). [6], [7] investigate deep learning based algorithms using large-scale radio dataset. They show that a multi-output classification model processing instantaneous radio measurements from specific 3GPP radio signals, as $RSRP$ or Timing Advance (TA) signals among other signals, achieves high accuracy for a relatively complex environment classification, considering multiple environments. Unlike above, there are only a few works where user environment inference is performed by exploiting or learning on time series data. They investigate mainly models for device. In the majority of them, the intention is to manually compute derived features and then use thresholds [2] or machine learning techniques to detect the environment [13], [14]. In [2], they classify the environment only in outdoor situation (pedestrian, in-car or non-moving).

To the best of our knowledge, our work is the first to perform user's environment detection, using LSTM-based classification algorithms combined with a heuristic related to user behavior. The IOD detection is done on network side by learning on multivariate time series of 3GPP radio data which, as per the standards, are supposed to be already measured by the network. Thus, we use models for detecting whether user will be indoor or outdoor at a given time.

III. Environment detection using time series data

In this section, we explain our approach. First, we start by describing our data. We then expound the motivations behind the proposed approach and then detail our proposal.

A. Dataset description

1) Data collection representative of real life: The data has been collected during the day and night and 7/7 days, during various activities of mobile users: while being static, while moving with different speeds and at various diverse

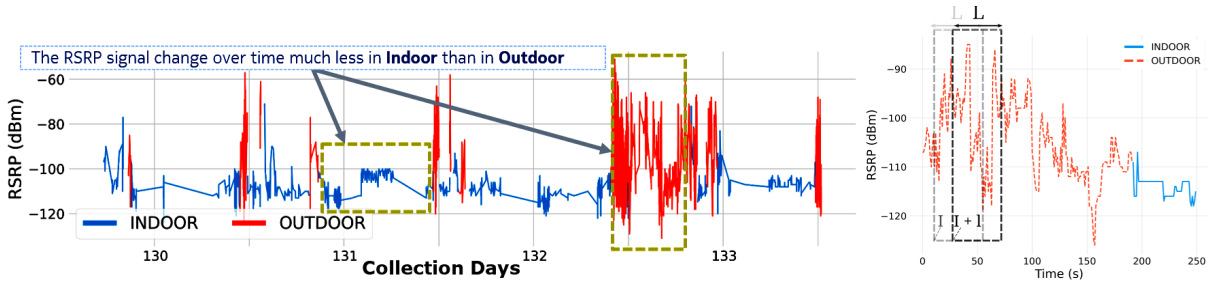


Fig. 1: a) Temporal variations of measured $RSRP$ (dBm) over time - b) Data sequence. Sliding window algorithm

cities/places. The data collection was conducted by volunteers. Their characteristics are detailed in Table I. The volunteers are urban and they have regular but specific lives and habits regarding the sites they crossed. The collection was done in several locations in France which were visited by the volunteers. Part of data has been labelled manually over an accumulated period of more than 200 days. After data cleaning and pre-processing phase, this enables us to build a representative dataset for training and evaluation which is as close as possible to the complexity and the usage situations variety of mobile phones and of users' movements in real world. So, the dataset used for training and evaluation is composed of many various instances of UE-specific 4G data (as 5G was not available at that time).

Data & Period	Location	User	I/O
$130K$ over 15 months	Urban, indoor, beach, mountains	Regular habits	Indoor: 63% Outdoor: 37%

TABLE I: Data collection configuration

2) Data Features: For environment detection, datasets are composed of some 3GPP standard radio features already employed in [6] and re-explained below. But, we target to investigate three features related to power, location and mobility signals. They are Key Performance Indicators and commonly used in mobile networks context and reflecting the behavior of users. Moreover, they contribute hugely to efficiently classify the ambiguous points and have shown significant impact on the model accuracy. The radio features are as follows:

- $RSRP$: average received power of a single Reference Signal (RS) resource element [15].
- TA: used to control UL signal transmission timing [16].
- MI: Mobility indicator refers to number of Cell ID changes in a sliding window of a given duration T_m [17]. To estimate its value, T_m is fixed to $100s$ [18].

TA is an indicator of the distance between a UE and a base station, whereas MI values depend on the mobility state. The information of distance combined with mobility information helps to distinguish measurement points, which have low $RSRP$ values even when they do not correspond to indoors. For example, when a user is moving at high speed, the radio signal quality can degrade (which is also the case when the user is indoors). As we are dealing with supervised methods, environment labels are used to train the models. Data cleaning and verification methods as described in [6] are applied to correct as much erroneous environment labels as possible. Thus, our whole data-set is composed of a vector of 3 features ($RSRP$, TA, MI) plus labels.

B. Learning on time series data

The measurements in the datasets have been collected in various situations and over a long period of 15 months. They also contain some breaks over time in data collection. The duration of these interruptions is up to several days (up to 30 days). Except these interruptions, the collection proceeded in an almost regular manner. Fig. 1 shows the temporal variations in $RSRP$ signal measurements. The blue curve corresponds to times when user was indoor, while the orange curve corresponds to times when user was outdoor.

As seen in Fig. 1a), the amount of $RSRP$ signal variations over time is different for indoor from that of outdoor. We observe also that $RSRP$ values vary less when user is indoor, while they vary much more when user is outdoor. The standard deviation is calculated from the whole time-series during data collection. Comparing RSRP standard deviation of both environment, we notice that the outdoor standard deviation (12.8dB) is almost 2.2 times higher than indoor standard deviation (5.8dB). This can be explained as follows. When user is outdoor and moving then the multiple paths of radio propagation can change, causing the phenomenon known as multi-path induced fading creating constructive or destructive interference as user is moving. In comparison, in indoor places, the users are relatively less mobile. Additionally, the peak $RSRP$ value can be relatively high in outdoor as compared to indoor. This is because the walls attenuate the radio signals more when user is indoor. These differences are more visible if we look at the sequence of temporal data. Using data sequences as input also indirectly adds information about the user's previous environment. Moreover, since users are more likely to stay in the same environment for some time, previous environments can also help to predict the next environment. There may exist some ambiguous points where some radio signal values corresponding to outdoor are similar to those of indoor, but have different variations over time. In addition, time series will help us distinguishing them.

Histogram in Tab. II depicts the empirical cumulative distribution of sojourn time per environment. It is computed on the whole data of volunteers. From the curve, we can derive the minimal and the maximal duration of stay in an environment before moving to another one. During the data collection period, we observed that the volunteers stayed in

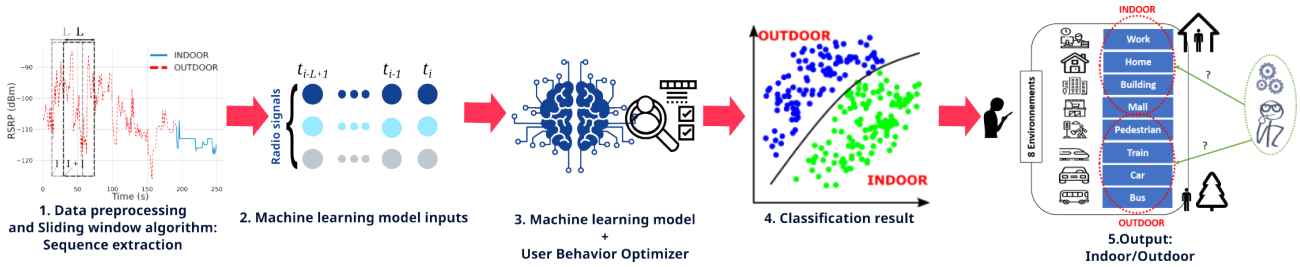
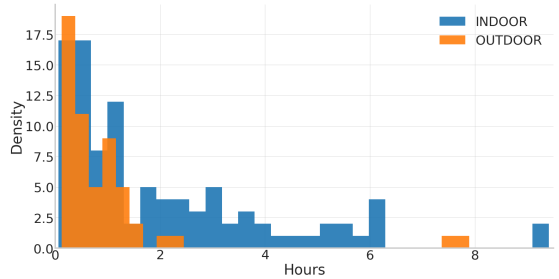


Fig. 2: User Environment Detection: step 1 to step 5 for 8 typical Indoor/Outdoor real-life environments

same environment with a minimal duration of 3m42s and a maximal duration of 9h25m indoor, where humans stay for longer time by nature. When we analyse their behavior more with statistics of sojourn time given in Table II, we observe a divergence of volunteers' behavior between indoor and outdoor environments. Statistically, Users remained in-



Sojourn Time	Min.	Mean	Median	75 % percentile	Max.
Indoor	3m42s	2h00m	1h11m	2h56m	9h25m
Outdoor	7m24s	57m	35m	1h05m	7h54m

TABLE II: Sojourn time statistics in I/O environment

doors longer than outdoors. However, the sojourn time is minimal indoor. As a matter of fact, this time value is a crucial information because it would fix the maximal upper limit of time series duration for accurate UED. Indeed, this constraint would help prevent an overlap of input sequences data from two cumulative transient periods, between different environments in the same processing window. Nevertheless, this duration should also be large enough to allow models to better recognize indoor or outdoor signatures over time, for correct environment detection.

IV. Time Series-based Indoor-Outdoor detection

Fig. 2 depicts our global approach followed to achieve time series-based ML techniques, for user environment detection. **Pre-processing:** The collected data is noisy. This is due to the use of a collection mode and the not 100% accurate of labeling by volunteer. In real world, users may face some software/hardware failures or disconnection events that may stop the data collection or introduce noise to the recorded values. The records that contain these noises are discarded if the failure was for a long period. When the failure was for a short period, causing only a few missing values, the missing values were interpolated.

Data sequence extraction for learning: This step is done by using a *sliding window* which starts by extracting the first data sequence of duration L of $RSRP$, TA and MI starting from an initial index I at the beginning of data. The index is then shifted by a stride l and used to extract the next sequence. The process is repeated until all the sequences are extracted. Extracted sequences of all the above features are then fed as input to ML model. Fig. 1 b) shows an iteration of sliding window algorithm in black applied on $RSRP$ for $L = 100s$ and a shift of $l = 1s$.

Machine learning algorithms: The classification problem is formulated as follows: Given a set of data denoted as: $\mathcal{D} = \{(t_i, \mathbf{x_1}, y_1), \cdots, (t_n, \mathbf{x_n}, y_n)\}$ where t_i is the i-th timestamp, $\mathbf{x_i} \in \mathbb{R}^m$ is the i-th data vector with m features, and $y_i \in \{1, \cdots, C\}$ is its label, the objective of time series classification is to predict class y_i from a sequence $\{\mathbf{x_{i-L}}, \cdots, \mathbf{x_i}\}$. For time series classification, we investigate multivariate LSTM algorithm which is well known due to its ability to efficiently extract useful information from sequential data. We also compare it with other DL architectures, such as FNN and Convolutional Neural Networks (CNN), in addition to other classical ML algorithms shown in Table III. The hyper-parameters listed in Table III will be discussed later in Section V-A. A baseline threshold-based algorithm that computes statistical features and achieves thresholding, is also tested. We consider that multivariate LSTMs to perform the best since its structure is designed for this kind of sequential data. LSTMs have recurrent connections which enable them to process sequences of data while treating each point in relation with the whole sequence. It retains useful information about previous data to help with the processing of new data points.

User behavioral optimizer: In addition to ML methods of classification, we introduce a module based on domain knowledge, which we call *User Behavior Optimizer*. Its aim is to leverage behavioral knowledge about users' movements

Algorithm	Optimized Hyper-parameters
K-NN [19]	• Number of neighbors: 5
SVM [20]	• Regularization parameter: 3 • Kernel: Radial basis function
Random Forest [20]	• Number of trees: 80 • Maximum depth of the tree: 16
AdaBoost [20]	• Base estimator: Decision Tree • Number of estimator: 50 - Learning rate: 1.0
FNN [21]	• Hidden layers: 4 layers - Dropout rate: 0.4 • Optimizer: Nadam - Loss: Binary cross entropy
CNN [22]	• Hidden layers: 5 layers - Dropout rate: 0.25 • Optimizer: Nadam - Loss: Binary cross entropy
LSTM [8]	• Hidden layers: 2 layers - Dropout rate: 0.25 • Optimizer: Nadam - Loss: Binary cross entropy

TABLE III: Hyper-parameters for supervised methods

in their environment to further improve user environment classification performances. An improvement is expected after correcting supposedly aberrant patterns output by the model. The goal of UBO is to detect and correct them when they do not match a real user behavior. For example, users usually do not change their environment twice during a short period. If we denote by τ_{max} the maximal upper limit of this short period, we can reasonably assume that τ_{max} is around $60s$. We propose two ways of utilising UBO for UED. The first one, referred as UBO-int, acts directly inside the model during training phase and the second one, referred as UBO-ext, acts after model output, see Fig. 3. The first step of both UBO-int and UBO-ext is to find aberrant user patterns (**AUP**) where the ML model outputs a user leaving and getting back to the previous environment. The procedure AUP is described as follows. As shown in Alg. 1, it outputs S_k^+ and S_k^- for environment k $\forall k \in \{1, \cdots, C\}$. We use S_k^+ to denote the set of indexes when a given user enters the environment k. We use the indicator function $\mathbb{1}_k(\hat{y}_i)$ such that it is 1 if $\hat{y}_i = k$, but 0 otherwise. Note that $\neg$ is used to denote negation. Thus, $\forall j \in S_k^+, \mathbb{1}_k(y_j) = 0$ and $\mathbb{1}_k(y_{j+1}) = 1$. We use S_k^- to denote the set of indexes when a given user leaves the environment k. Thus, $\forall j \in S_k^-, \mathbb{1}_k(y_j) = 1$ and $\mathbb{1}_k(y_{j+1}) = 0$.

We intend to inspect the sequence of predicted labels $\{\hat{y}_1, \cdots, \hat{y}_l\}$ to look for anomalous patterns and we try to correct them. Lets $o_i^{(k)}$ be the softmax output of the neural network model corresponding to the environment k and $\hat{y}_i = \arg\max_{k \in \{1,\cdots,C\}} o_i^{(k)}$ represents the predicted class of the i-th example. These predictions $\hat{y}_i$ need to be corrected.

Algorithm 1: Find Aberrant User Patterns (AUP)

Input: $\{\hat{y}_{i-L}, \cdots, \hat{y}_i\}$
Output: $\{S_k^+, S_k^-\}$ $\forall k$
Iterate over all predicted labels and classes.
foreach $k \in \{1, \cdots, C\}$ **do**
  Initialize $S_k^- = \{\}$, $S_k^+ = \{\}$
  foreach $j \in \{i - L \,..\, i\}$ **do**
    if $\mathbb{1}_k(\hat{y}_j) \wedge \neg\mathbb{1}_k(\hat{y}_{j+1})$ **then**
      $S_k^- \leftarrow S_k^- \cup \{j\}$
    else if $\neg\mathbb{1}_k(\hat{y}_{j-1}) \wedge \mathbb{1}_k(\hat{y}_j)$ **then**
      $S_k^+ \leftarrow S_k^+ \cup \{j\}$
    continue;
  continue;

UBO-int acts internally within the model, as described in Alg. 2. It will penalize instances when l successively detected environment classes contain anomalous patterns. For this, instead of using classic loss function, we introduce a new additive term $\alpha\ \mathcal{L}_{UBO}$ in the loss function. $\mathcal{L}_{UBO}$ is a differentiable function that penalize sequences containing the aberrant user behavior and $\alpha \leq 1$ is a weight factor. Note that to be able to do as above, we use LSTM model which outputs data sequences instead of just a single element. $\mathcal{L}_{UBO_i}$ is estimated as follows. We first estimate δ_k for each class k:

$$\delta_k = \sum_{j^+ \in S_k^+} \sum_{j^- \in S_k^-} (1 + |o_{j^+}^{(k)} - o_{j^-}^{(k)}|) \times (t_{j^+} - t_{j^-}) \quad (1)$$

Then we use δ_k to compute:

$$\mathcal{L}_{UBO_i} = \frac{1}{C \times L} \sum_{k=1}^{k<C} \frac{1}{1 + \delta_k} \quad (2)$$

UBO-ext described in Alg. 3 is an external system interfaced with LSTM model output. It monitors sequences of l successively detected classes of environment to spot as well as correct the aberrant patterns. The detected patterns must also be of short period $< \tau_{max}$ with the mean of confidence scores of these prediction, computed by the model softmax output $(\mathbf{o}_i)$, $< C_{th}$. The corresponding instances are corrected by setting their predicted classes to previous environment. τ_{max} and C_{th} are learned during the training step to maximize number of corrected instances by UBO-ext while reducing potential added errors by having a big τ_{max}. Fig. 4 shows an example sequence where UBO-ext corrected false patterns.

Algorithm 2: User Behavior Optimizer Internal (UBO-int)

Input: $\{\hat{y}_{i-L}, \cdots, \hat{y}_i\}$, $\{\mathbf{o}_{i-L}, \cdots, \mathbf{o}_i\}$, $\{t_{i-L}, \cdots, t_i\}$
Output: $\mathcal{L}_{UBO_i}$
Find Aberrant Patterns.
$S_k^+, S_k^- \leftarrow$ AUP($\{\hat{y}_{i-L}, \cdots, \hat{y}_i\}$)
Estimate δ_k and then L_{UBO_i} using eq.(2).

One has to notice that the actions of UBO-int and UBO-ext are different. Nevertheless, both of them follow the same objective: detecting behavioral aberrations at sequence level and then acting to correct them.

Algorithm 3: User Behavior Optimizer External (UBO-ext)

Input: $\{\hat{y}_{i-L}, \cdots, \hat{y}_i\}$, $\{\mathbf{o}_{i-L}, \cdots, \mathbf{o}_i\}$, $\{t_{i-L}, \cdots, t_i\}$
Output: Corrected predictions $\{\hat{y}_{i-L}, \cdots, \hat{y}_i\}$
Find Aberrant Patterns.
$S_k^+, S_k^- \leftarrow$ AUP($\{\hat{y}_{i-L}, \cdots, \hat{y}_i\}$)
Correct previously predicted labels.
foreach $j^- \in S_k^-, j^+ \in S_k^+$ **do**
  $\Delta t \leftarrow t_{j+} - t_{j-}$
  $\mu_c \leftarrow \frac{1}{j^+ - j^- - 2} \sum_{j=j^-+1}^{j^+-1} o_k^{(j)}$
  if $\Delta t < \tau_{max} \ \wedge \ \mu_c < C_{th}$ **then**
    $\hat{y}_{j^-+1:j^+-1} \leftarrow \hat{y}_{j^-}$

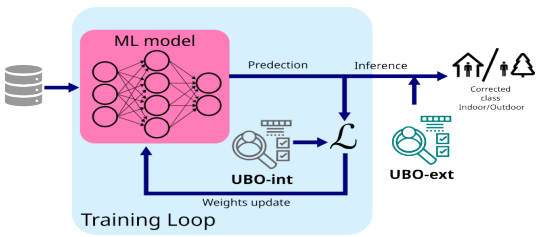


Fig. 3: UBO-int and UBO-ext location

Performance evaluation To avoid data leakage while training on time series data and in order to fairly evaluate the models, we use two different validation methods. The first

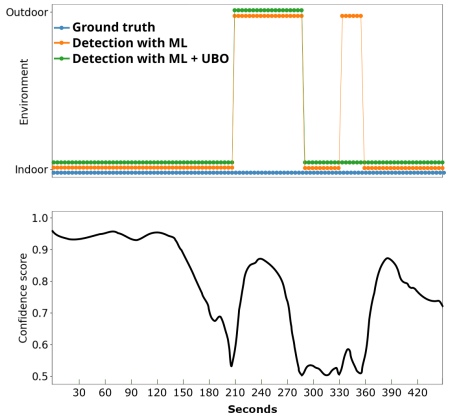


Fig. 4: User behavioral optimizer: environment output and confidence score

method that we refer as *Time-Block validation* method consists in slicing the data into blocks, using the data collection interruption times as boundary markers. Then, each block is randomly assigned to either training or test set. These boundaries caused by data interruptions ensure that there is no correlation in time between the training dataset and the test dataset. The second one is the *Forward-chaining cross-validation* method also known as rolling-origin-recalibration evaluation in [23]. Next section evaluates the performance of several environment detection algorithms using both these validation methods over multiple experiments. The investigation is conducted on the aspects of time series and the impact of UBO. The model evaluation is done with optimal values of confidence threshold.

V. Results and Discussions

In this section, we first describe the configuration setup. We then discuss our experiments and the results obtained for Indoor/Outdoor detection using time series and UBOs.

A. Configuration setup

Before training the classifiers, we started by tuning the hyper-parameters for each ML algorithm. FNN, CNN, and LSTM model hyper-parameters were optimized using Bayesian optimization, while those of other algorithms were manually tuned in Table III. We used 70% of the labeled data for training and the remaining 30% for evaluation. As mentioned in Section III-B, we use two evaluation methods. Note that we split the datasets into different blocks. Then for each *experiment*, the blocks are randomly assigned to either the training or the test sets. We generate a total of 10 and 5 experiments for Time-Block validation and Forward-chaining cross-validation method, respectively. Our choice is motivated by the need to ensure sufficient sizes for training and test blocks. For performance evaluation of models, we also compute average of balanced Accuracy and $F1-score$ that is expressed as: $F1-score = \frac{1}{N}\sum_i^N 2.\frac{Precision_i \times Recall_i}{Precision_i + Recall_i}$, where $N = 2$ is number of classes. Precision is number of correct positive results divided by number of all positive results returned by the classifier, and Recall is number of correct positive results divided by number of all relevant samples. Optimizations and experiments, including training and evaluations, are performed on a Linux machine equipped with Intel Core i7-5930 CPU and GPU, Nvidia GTX Titan X graphic card and 64 GB of RAM. Neural network models are trained on 1 GPU while other models are training on 1 CPU.

B. Machine learning performance for Time Series

Table V shows the performance of environment detection algorithms with both validation methods. To investigate the impact of time series, we trained multiple models using different duration values, L, of sliding window as input. We studied the cases for: $L = 1\ sample$, $L = 90s$ and $L = 190s$. The case "$L = 1\ sample$" corresponds to only using 1 instantaneous value and not a data sequence. Note that, as discussed in III-B, we limit $L \leq 3m42s$ where $3m42s$ is the minimal sojourn time in an environment. As we are interested in limiting the detection delays, hence we do not test higher durations than $190s$, namely $3min10s$.

Lets first see the performance of ML models with $L = 1\ sample$, i.e, only where '1 samp.' option is present in the table. The best among them is Adaboost with an $F1-score$ of 88.8% for Time-Block and with an $F1-score$ of 88.6% for Forward-chaining cross-validation. When we exploit time-series data, the best $F1-score$ is shown by LSTM with $L = 190s$, which is 94.4% and 91.4% for Time-Block and Forward-chaining validation, respectively. We also note good performances shown by Random Forest and CNN. CNN's good performance is because the 1D CNN is able to consider time-series data. For all algorithms, except FNN in some cases, the models which use data sequences outperform up to around 4% the ones that only use a single measurement. The above is observed in most cases, for both Time-Block validation and Forward-chaining cross-validation. Further, we can observe that for SVM, Random Forest, Adaboost and FNN with $L = 90s$ perform better than when using $L = 190s$ in certain cases. While LSTM and CNN perform the best with $L = 190s$. This may mean that beyond a certain duration of sliding window, measurements that are too old are irrelevant for some machine learning classifiers while LSTM and CNN are still able to exploit them.

Overall, LSTM with $L = 190s$ with $F1-score = 94.4\%$ for Time-Block has the best performance, in almost all experiments, as compared to other models (as seen in Table V).

C. Impact of UBO on machine learning model performance

How do UBOs act?

We conduct a finer analysis using LSTM with $L = 190s$ as well as UBOs with $\tau_{max} = 60s$ and $\alpha = 0.3$ for a given C_{th}. The performance is evaluated over 10 experiments with Time-Block validation. This allows us to understand how UBOs act. The role of UBOs is to detect, avoid or correct unwanted patterns. A pre-analysis of the ground-truth dataset showed us that initially it did not contain any unwanted patterns. Table IV shows the number of unwanted patterns (UPs) for different UBO scenarios in case of the 8th experiment (with $C_{th} = 0.9$).

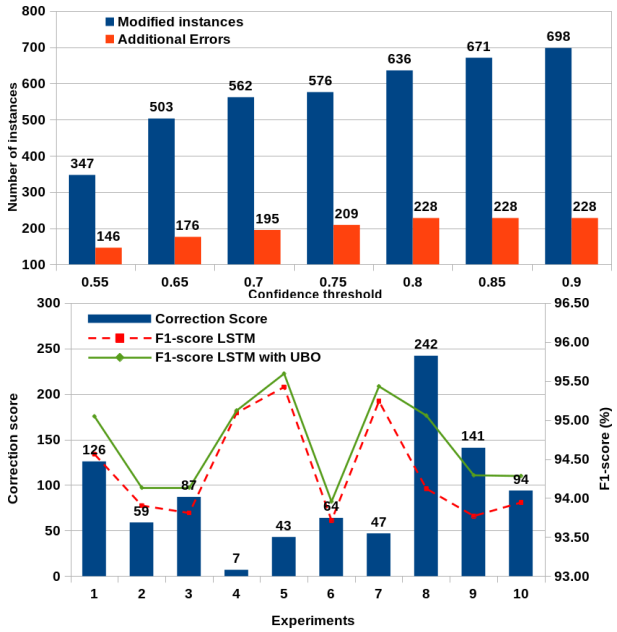


Fig. 5: Case of LSTM and $L = 190s$: a) Modified instances (blue) and additional errors (orange) versus confidence threshold - b) Correction score (blue), F1-score without (orange) and with User Behavioral Optimizer (green) versus experiment

As shown in Table IV, we remark that the output of LSTM has some UPs within predicted classes. We can also observe that enabling UBOs permits us to remove more than half of the anomalies. To continue the analysis, Fig. 5 a) illustrates the impact of UBO-ext for different confidence thresholds. The figure shows the evolution of the total number of modified instances (blue) as well as the additional errors (AEs) (orange) when UBO-ext is enabled. The additional errors represent the predicted classes which were correctly detected by LSTM initially, but correction by UBO-ext made them erroneous. We observe that both the numbers increase with C_{th}. However, the error growth saturates and quickly reaches a maximal value of 228 errors at $C_{th} = 0.7$. In contrast, the modified instances continue to grow and reach a maximal number of 698 instances at $C_{th} = 0.9$. At this threshold, UBO-ext is still able to positively correct the instances without introducing AEs. We define the correction score (CS) as the difference between the number of correctly corrected instances and AEs. Fig. 5 b) depicts $F1 - score$ and CS for the 10 different experiments. For that we did several experiments after configuring UBO-ext with optimal values of C_{th} which maximize CS. In the figure, CS varies between 7 and 242 instances correctly modified. We note that $F1 - score$ (green curve) gets enhanced in each case by using UBOs. Thus, these experiments demonstrate that proposed UBOs can cope with various situations having different amounts of errors to correct.

	w/o UBO-int & UBO-ext	w/o UBO-int & w. UBO-ext	w. UBO-int & w/o UBO-ext	w. UBO-int & UBO-ext
UPs	101	53	62	23

TABLE IV: UBOs impact on the unwanted patterns (UPs)

Performance enhancement:
Experiments done with the other ML algorithms show a very small $F1 - score$ improvement around $+0.2\%$. The poor gain obtained can be explained by their internal structure which did not allow to integrate UBO-int inside their model. Thus, in terms of best result obtained among other ML algorithms, $F1 - score$ of SVM increases to maximal value of 92.0% with UBO-ext and $L = 190s$. In contrary, UBOs improve LSTM's performance globally by $+0.8\%$ as compared to LSTM without UBOs. Overall, with UBOs and $L = 190s$, LSTM remains the best with mean $F1 - score$ of 94.8% for Time-Block validation and 92.2% for Forward-chaining cross-validation, as compared to other models (as seen in Table V). Similar trend is observed with $Accuracy$. Note that for CNN's $F1 - score$ increases by $+0.55\%$. This is mainly thanks to UBO-ext, since adding UBO-int degrades the performance.

Trade-off between $F1 - score$ and time series duration:
Experiments demonstrate that training ML classification models over time series, for UED performance, is beneficial. Nevertheless, using long time series data also has some drawbacks. A high value of L can delay the detection of environment change. Moreover, it also increases the amount of historical data to be stored for performing detection. For DL models, a high value of L leads to model complexity. The training execution time increases with L as it can be seen in Table V. Note that hyper-parameters optimization is done beforehand such that the training execution time doesn't include the hyper-parameters optimization phase. However, we can notice that adding UBOs doesn't increase the training execution time a lot while it still improves the overall performance. Thus, using UBOs will give us an opportunity to have a trade-off between $F1 - score$ and L, while choosing an appropriate model.

Algorithms	Sliding Window (seconds)	Accuracy (%) Time-Block	Accuracy (%) Fwd-chaining C-validation	F1-Score (%) Time-Block	F1-Score (%) Fwd-chaining C-validation	Training Time (s)
Threshold-based Model	90s	81.3	84.5	81.2	83.2	0.08
	190s	**86.0**	**87.2**	**86.1**	**85.7**	**0.13**
K-NN	90s	86.8	**87.4**	86.8	**86.8**	**423.5**[1]
	190s	**87.5**	87.3	**87.6**	86.5	743.8[1]
SVM (RBF kernel)	1 samp.	86.5	88.6	87.2	87.6	24.5
	90s	90.7	90.2	91.3	89.5	71.9
	190s	**91.6**	**90.4**	**92.0**	**89.7**	89.5
Random Forest	1 samp.	88.0	86.5	87.7	85.4	2.2
	90s	**91.9**	90.4	**92.0**	90.1	7.7
	190s	91.7	**91.1**	91.7	**90.7**	**11.2**
AdaBoost	1 samp.	88.4	89.4	88.8	88.6	0.8
	90s	**90.9**	90.6	**91.2**	90.4	**9.1**
	190s	90.0	**91.1**	90.3	**90.8**	18.7
FNN	1 samp.	87.2	**87.3**	87.4	85.8	28.6
	90s	**90.7**	86.5	**91.2**	**86.3**	**30.4**
	190s	91	83.8	91.5	83.2	33.8
CNN	90s	92.2	90.3	91.7	90.7	37.6
	190s	**92.7**	**91.1**	**92.1**	**91.3**	**42.8**
with UBO-ext	90s	92.8	91.1	92.3	91.5	38.8
	190s	**93.2**	**91.4**	**92.7**	**91.6**	**44.9**
with UBO-int&ext	90s	91.4	**88.6**	92.0	**87.9**	43.1
	190s	**91.7**	88.4	**92.2**	86.1	**48.6**
LSTM	90s	92.9	91.3	93.3	90.1	68.7
	190s	**94.2**	**92.4**	**94.4**	**91.4**	**80.2**
with UBO-ext	90s	93.8	91.7	94.1	90.7	70.9
	190s	**94.6**	**93.1**	**94.7**	**92.1**	**82.4**
with UBO-int&ext	90s	93.8	91.8	94.1	90.7	72.3
	190s	**94.8**	**93.2**	**94.8**	**92.2**	**89.6**

[1] The times reported for K-NN are instead the times taken for classification. The classification times for other ML algorithms were negligible.

TABLE V: Comparison of supervised ML methods: Accuracy, $F1 - score$ and Training Time vs. sliding window duration - Time-Block and Forward-chaining cross-validation methods

The baseline model is fastest to train. Apart from the baseline model, Random Forest and AdaBoost are, relatively, the fastest to train, except for SVM. Random Forest is the fastest when using time series data. It takes around $8s$ but delivers a lower mean $F1-score$ of 92%. LSTM models, which are the best performing models, are among the slowest. Without UBOs and $L = 190s$, LSTM takes up to the maximal time $80.2s$ whereas, with UBOs , LSTM takes up to $89.6s$, which is the highest time among all the tested models. But, with UBOs and $L = 90s$, LSTM takes up to $72.3s$ and delivers a mean $F1-score$ of 94.1%, which is faster, but with slightly less $F1-score$ than those with $L = 190s$. Thus, LSTM model with $L = 90s$ and UBOs remains eligible for selection as it is faster for offline or real time training. Note that K-NN takes the longest time. Practically, the time complexity of K-NN for training is $O(1)$ while it is $O(n)$ for classification depending upon the number of examples.

Finally, our experiments also show the benefit of using domain knowledge in the form of UBOs.

VI. Conclusion

In this paper, we have investigated machine learning based Indoor/Outdoor detection methods by processing 3GPP radio data and exploiting sliding sequences of multivariate time series data. The variations of radio signals over time are clearly different when user is located indoor as compared to that of outdoor. In this work, the experiments have shown that machine learning models using time series perform the best for Indoor/Outdoor detection in real-life situations.

Then, we proposed to use domain knowledge based module, called User Behavioral Optimizer (UBO), to improve Indoor/Outdoor detection. Experiments demonstrate the benefit of using UBOs with LSTM models for real-time user environment detection. We discussed how UBOs can improve the performance for some time-series based LSTM models. UBOs can improve the performance of a model using shorter time series, such that, it comes close the performance of the model with longer time series. Using shorter time series has following advantages: it is faster to train and to detect class changes, requires lower storage space while classification and is relatively less complex. Thus, thanks to UBO we get more choices in machine learning model selection, with reasonable trade-offs in terms of accuracy, speed and storage. Therefore, LSTM with UBOs can be a good solution for sensing environmental situations of mobile users within network infrastructure.

In future research, we plan to explore other machine learning algorithms/architectures that are more robust to interruptions and irregularly spaced time series. Moreover, we intend to explore the proposed UBO algorithm with multi-class classification. Indeed, with higher number of classes, the classification output might oscillate erroneously between multiple classes. We would like to study whether UBO is able to detect and correct such anomalous behaviour.

References

[1] A. Ray, S. Deb, and P. Monogioudis, "Localization of LTE measurement records with missing information," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[2] A. B. H. Alaya-Feki, A. Le Cornec, and E. Moulines, "Optimization of radio measurements exploitation in wireless mobile networks." *JCM*, vol. 2, no. 7, pp. 59–67, 2007.

[3] E. Pateromichelakis, F. Moggio, C. Mannweiler, and al, "End-to-end data analytics framework for 5g architecture," *IEEE Access*, vol. 7, 2019.

[4] S. Mekki, T. Karagkioules, and S. Valentin, "HTTP adaptive streaming with indoors-outdoors detection in mobile networks," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2017, pp. 671–676.

[5] J. Faouzi, "Time series classification: A review of algorithms and implementations," in *Machine Learning (Emerging Trends and Applications)*. Ketan Kotecha / https://hal.inria.fr/hal-03558165, 2022.

[6] I. Saffar, M. L. A. Morel, K. D. Singh, and C. Viho, "Semi-supervised deep learning-based methods for indoor outdoor detection," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7.

[7] I. Saffar, M. L. A. Morel, M. Amara, K. D. Singh, and C. Viho, "Mobile user environment detection using deep learning based multi-output classification," in *2019 12th IFIP Wireless and Mobile Networking Conference (WMNC)*. IEEE, 2019, pp. 16–23.

[8] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[9] A. Diamanti, J. M. S. Vilchez, and S. Secci, "Lstm-based radiography for anomaly detection in softwarized infrastructures," in *2020 32nd International Teletraffic Congress (ITC 32)*. IEEE, 2020, pp. 28–36.

[10] P. Zhou, Y. Zheng, Z. Li, M. Li, and G. Shen, "Iodetector: A generic service for indoor-outdoor detection," in *Proceedings of the 10th acm conference on embedded network sensor systems*, 2012, pp. 113–126.

[11] J. Bejarano-Luque, M. Toril, M. Fernandez-Navarro, R. Acedo-Hernandez, and S. Luna-Ramirez, "Data-driven algorithm for indoor-outdoor detection based on connection traces in a lte network," *IEEE Access*, vol. 7, p. 65877–65888, 2019.

[12] C. Frank, R. S. Phillipson, I. Wezeman, and Chiscop, "Indoor–outdoor detection in mobile networks using quantum machine learning approaches," *Computers*, 2021.

[13] A. Esmaeili Kelishomi, A. Garmabaki, M. Bahaghighat, and J. Dong, "Mobile user indoor-outdoor detection through physical daily activities," *Sensors*.

[14] Y. Zhu, H. Luo, F. Zhao, and R. Chen, "Indoor/outdoor switching detection using multisensor densenet and lstm," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1544–1556, 2020.

[15] *Evolved Universal Terrestrial Radio Access (E-UTRA); Requirements for support of radio resource management*, 3GPP Standard TS 36.133 - Release 8, 2018.

[16] *Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification*, 3GPP Standard TS 36.321- Release 15, 2018.

[17] *Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) procedures in idle mode*, 3GPP Standard TS 36.304 - Release 15, 2018.

[18] D. Herculea, C. S. Chen, M. Haddad, and V. Capdevielle, "Straight: Stochastic geometry and user history based mobility estimation," in *Proceedings of the 8th ACM International Workshop on Hot Topics in Planet-scale mObile computing and online Social neTworking*, 2016.

[19] Z. Zhang, "Introduction to machine learning: k-nearest neighbors," *Annals of translational medicine*, vol. 4, no. 11, 2016.

[20] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. " O'Reilly Media, Inc.", 2019.

[21] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[22] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, no. 1, pp. 1–74, 2021.

[23] C. Bergmeir and J. M. Benítez, "On the use of cross-validation for time series predictor evaluation," *Information Sciences*, vol. 191, pp. 192–213, 2012.