This is the author's accepted version of the article. The final version published by IEEE is S. Bakirtzis, K.Qui, I. Wassell, M. Fiore and J. Zhang, "Deep Learning-based Multivariate Time Series Classification for Indoor/Outdoor Detection" IEEE Internet of Things Journal, vol TBD, pp. TBD, DOI: 10.1109/JIOT.2022.3190555.

Deep Learning-based Multivariate Time Series Classification for Indoor/Outdoor Detection

Stefanos Bakirtzis, *Member, IEEE*, Kehai Qiu, *Student Member, IEEE*, Ian Wassell, Marco Fiore, *Senior Member, IEEE* and Jie Zhang, *Senior Member, IEEE*

***Abstract*—Recently, the topic of indoor outdoor detection (IOD) has seen its popularity increase, as IOD models can be leveraged to augment the performance of numerous Internet of Things and other applications. IOD aims at distinguishing in an efficient manner whether a user resides in an indoor or an outdoor environment, by inspecting the cellular phone sensor recordings. Legacy IOD models attempt to determine a user's environment by comparing the sensor measurements to some threshold values. However, as we also observe in our experiments, such models exhibit limited scalability, and their accuracy can be poor. Machine learning (ML)-based IOD models aim at removing this limitation, by utilizing a large volume of measurements to train ML algorithms to classify a user's environment. Yet, in most of the existing research, the temporal dimension of the problem is disregarded. In this paper, we propose treating IOD as a multivariate time series classification (TSC) problem, and we explore the performance of various deep learning (DL) models. We demonstrate that a multivariate TSC approach can be used to monitor a user's environment, and predict changes in its state, with greater accuracy compared to conventional approaches that ignore the feature variation over time. Additionally, we introduce a new DL model for multivariate TSC, exploiting the concept of self-attention and atrous spatial pyramid pooling. The proposed DL multivariate TSC framework exploits only low power consumption sensors to infer a user's environment, and it outperforms state-of-the-art models, yielding a higher accuracy combined with a smaller computational cost.**

***Index Terms*—indoor outdoor detection, deep learning, time series classification, self-attention, seamless navigation.**

I. Introduction

EMPOWERED by the major advances in sensor, computing, communication and networking technologies, the Internet of Things (IoT) has flourished over the last decade or so [1]. The unprecedented number of mobile and sensing devices produces an enormous volume of data, which can be exploited for the development of intelligent interconnected environments and the design of new smart applications [2]. Among the various IoT applications, location-based services and context-aware applications have received strong attention across the research community [3]–[6].

Such applications adapt their behaviour based on contextual information, such as the user's profile, their location, and their surroundings, along with others. Their design and performance can be enhanced significantly by characterising a user's environment, i.e., whether a user is found in an indoor or in an outdoor space. Knowledge of a user's environment has been recently leveraged to develop healthcare applications [7], [8], design seamless navigation systems [9]–[11], and augment the efficiency of wireless network operation [12], [13]. Hence, a significant effort has been made to develop accurate and efficient indoor outdoor detection (IOD) models [14]–[20].

Existing IOD models make use of the mobile device sensors to determine a user's environment, and they can be separated into two categories: (i) threshold-based and (ii) machine learning (ML)-based models. In the first case, the user's environment is determined by comparing measurements from the cellular phone sensors to some predefined fixed values [14], [16]. The fixed values are typically determined empirically through measurement campaigns. In the second case, the mobile phone sensor measurements are exploited to train ML algorithms, which learn to infer a user's environment [15], [17], [18], [20]. Threshold-based models are lightweight, easy to implement and interpret, however they lack scalability and their accuracy can deteriorate significantly in environments other than those in which the measurement campaigns take place [15].

Indeed, the idea of determining a user's environment based on some fixed threshold values, derived from a limited data set, is innately constraining and it cannot be compared with the flexibility of an ML-based IOD model where the weights are iteratively selected such as to maximize the IOD accuracy. Thus, ML-based modes are considerably more versatile, but they require a large and diverse quantity of training data for their performance to be satisfactory [18]. More importantly, once trained offline, ML-based models can be employed in real-time to infer a user's environment in a computationally efficient manner, since their predictions are based only on simple operations between matrices. The fact that the main computational burden of ML-based models resides in their off-line training stage, along with their augmented generalizability, has fostered their wider adoption, and hence they have lately attracted greater scientific interest [15], [17], [18], [20]–[25].

This work was supported by European Commission through the Horizon 2020 Framework Programme, H2020-MSCA-ITN-2019, MSCA-ITN-EID, Grant No. 860239, BANYAN.

Stefanos Bakirtzis, Kehai Qiu and Dr. Ian Wassell are with the Department of Computer Science and Technology, University of Cambridge, Cambridge, CB3 0FD, United Kingdom (e-mail: ssb45/kq218/ijw24@cam.ac.uk).

Dr. Marco Fiore is with the IMDEA Networks Institute, Madrid, Spain (e-mail: marco.fiore@imdea.org).

Professor Jie Zhang is with the Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield, S10 2TN, United Kingdom (e-mail: jie.zhang@sheffield.ac.uk).

Stefanos Bakirtzis, Kehai Qiu and Professor Jie Zhang are also with Ranplan Wireless Network Design, Upper Pendrill Court, Ermine Street North, Papworth Everard, Cambridge, CB23 3UY, United Kingdom. (e-mails: stefanos.bakirtzis/kehai.qiu/jie.zhang@ranplanwireless.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier: 10.1109/JIOT.2022.3190555.

The viability of ML-based IOD models has alredy been proved for several applications critical for the IoT ecosystem, such as seamless navigation systems and healthcare services [8], [9].

The preponderance of the current ML-based models treat IOD as a simple classification problem, categorizing users either as indoor or outdoor according to the instantaneous mobile sensor measurements, ignoring the temporal dimension of the problem [15], [17], [18]. However, a user's environment can often be classified more easily based on the sensor reading variation over time, rather than by only inspecting their instantaneous values. For instance, when a user enters a building it should be expected that the received cellular signal power will exhibit a rapid fall. Thus, lately there have been attempts to exploit the temporal nature of the problem, considering sensor information not only from the current state, but from previous states as well [19], [20]. In [20], a long short-term memory (LSTM)-based model is employed to determine a user's environment. Hence, instead of classifying user's environments based only on the current mobile sensor readings, the authors also considered measurements from previous time steps. Although the proposed model yields a high accuracy, it entails substantial complexity, much of which we prove to be redundant. This leaves the doors open for further research on more expedient models, which can demonstrate the same or higher accuracy in a more computationally efficient way. Furthermore, it is also necessary to ensure that the IOD models developed exhibit only a small delay when an environment transition occurs (e.g., moving from outdoors to indoors).

Apart from ensuring a high accuracy and small transition detection time, IOD models ought to be power-friendly, keeping the cell phone power consumption to a minimum. A lot of recent work on ML-based IOD has leveraged signals from the GPS or the WiFi sensors to determine a user's environment [11], [20], [22], [26]. In such cases, constant use of the GPS sensor or continuous WiFi scanning can provoke rapid battery drain. Thus, despite the accuracy of these models, the use of power-hungry sensors can put in jeopardy the efficiency of the IOD model. Instead, the exploitation of the cellular signal, which is always recorded by default by the mobile device operating system, can be considerably advantageous in terms of an IOD model's power efficiency. Additional low energy consumption sensors, such as the accelerometer or the light sensor readings, can be employed to augment the performance of the IOD model while preserving a low power consumption.

Hence, striking a balance between high IOD accuracy, fast environment transition detection, and a high computational and power efficiency still remains an open problem. Another significant problem related to the ML-based IOD models is their interpretability. Threshold-based models are easy to explain, since their environment decision is based on simple comparisons between the sensors' readings and some fixed values. On the contrary, the complex nature of ML algorithms makes it practically impossibly to understand the reasoning behind the predictions made by ML-based IOD models. Consequently, it is infeasible to either identify wrong predictions or to simplify and improve the power efficiency of the ML model by eliminating input features that have only a minor contribution to prediction accuracy.

In this paper we seek to address these challenges. Unlike previous ML-based models [15], [17], [18], [21], [23]–[25], we study IOD as a multivariate time series classification (TSC) problem, investigating the performance of different deep learning (DL) TSC models. We refrain from using power-hungry sensors, thus the multivariate time series comprise readings from sensors that are either always active or consume only a small amount of power. Additionally, we employ the concept of self-attention [27] for the first time in the context of IOD, enabling an increase of IOD accuracy by allowing a DL sequence model to focus on the most significant parts of the multivariate sequence. We then introduce a DL framework for multivariate TSC, consisting of cascaded convolutional blocks, and an atrous spatial pyramid pooling (ASPP) [28] layer, followed by a two-layer LSTM using self-attention [27]. Our results indicate that accounting for the sensor measurement variation over time can considerably augment the performance of an IOD model. Finally, we leverage recent research that helps the interpretability of ML algorithms [29], in order to evaluate the importance of the various sensors used, and make our framework more power-friendly.

To summarize, the contributions of this work are:

- We demonstrate that a multivariate TSC approach exhibits a higher accuracy compared to standard ML-based classification IOD models, and it also outperforms conventional threshold-based models.
- We employ an attention mechanism in order to identify the parts of the time series that contribute the most towards the characterisation of a user's environment. We also use a pre-processing layer, consisting of consecutive standard and atrous convolutions, that allows the initial time series to be transformed and to handle larger time sequences, without comprising the computational efficiency of the proposed model. The proposed DL framework outperforms existing TSC models, in terms of accuracy and computational efficiency, especially for large measurement sequences.
- To infer a user's environment the proposed model builds only on sensors with low power consumption, such as the cellular signal and the accelerometer readings. Moreover, to make the proposed model interpretable, we make use of the Shapley additive explanations (SHAP) framework [29]. This allows the most significant features to be identified and, if necessary, eliminate the features with the lower contribution, thus further decreasing the power consumption of our model.
- To make our work reproducible, and allow researchers to further advance the state-of-the-art in the future, we make our data available at a public repository [30].

The outline of this paper is as follows. First, in Section II, we survey the state of the art in IOD, presenting existing threshold-based and ML-based IOD models. Consequently, in Section III, we explain how we formulated our multivariate TSC approach, we briefly discuss some commonly used DL TSC models, we introduce a novel multivariate DL TSC model employed for IOD, and we explain the main idea behind the

SHAP framework. In section IV, we explore the performance of the various DL TSC and threshold-based models for IOD, comparing their accuracy, computational efficiency and how rapidly they can detect environment transitions. Moreover, we show how these metrics are affected as the size of the input multivariate time series increases, i.e., as we consider larger sequences of measurements for the IOD classification. Finally, Section VI concludes the paper by outlining its main contributions.

II. State of the Art

Indoor outdoor detection models exploit the measurements from the mobile phone sensors to characterize a user's environment. Typical features used include the GPS, the WiFI or the cellular signal, the ambient light and the magnetic intensity, and measurements from the inertial sensors. Each feature exhibits a different behaviour in indoor and outdoor environments, thus by inspecting their values one can infer a user's environment. Specifically, cellular and GPS signals undergo significant attenuation when they propagate through construction materials [26], [31]. Light intensity is expected to be lower inside buildings during daytime, while the opposite holds during the night [14]. The magnetic field demonstrates strong fluctuations in the presence of ferromagnetic materials and electrical devices, which are commonly found in indoor environments [14]. Finally, inertial sensors can be exploited to detect whether a user is in motion.

The topic of IOD was first studied in [14], where the authors used the cellular signal, the ambient light and the magnetic field intensity to distinguish indoor from outdoor users. The authors considered two cases: (i) stateless, where the environment was determined by the instantaneous sensor readings, and (ii) stateful, where the previous state (indoors, semi-outdoors or outdoors) was also considered during the current environment decision-making. In the first case, each feature was processed separately by a detector, which compared the measured feature values to a standard threshold value, and pronounced an environment type along with a confidence level associated with its decision. The final environment decision was made by selecting the environment with the highest confidence level. In the second case, a first-order hidden Markov model was employed to estimate the probability of being at each state at every time step, given the observed sensor measurements, the prior state, and the transition probability between the different states.

Another threshold-based approach was presented in [16], using as features the cellular and the WiFi signal, the ambient light intensity and the accelerometer measurements. The user environment was determined based on four conditional subsequent comparisons between the measured features and some fixed-values. First, the accelerometer measurements were used to determine the state of the user (in-vehicle, walking, standing), marking high-speed vehicle users as outdoors. If the user was standing or was in motion, then the cellular signal was employed for IOD, and in case there was still a high uncertainty, the final decision was made based on the WiFi signal strength. However, as mentioned earlier, it is unlikely that these models will be reliable in environments other than those in which they were developed [15].

Machine learning-based IOD models don't suffer from this limitation, as they can be trained with crowd-sourced data originating from many different areas. Such an approach was presented in [15], where IOD was studied as a supervised and semi-supervised learning problem. The authors considered five different ML models and explored their IO classification performance, demonstrating that an ML-based IOD approach outcompetes conventional threshold-based models. A similar approach was presented in [17], [18], where data from a mobile network operator was used to carry out IOD. The authors employed a self-supervising multilayer perceptron (MLP)-based scheme to classify the users' environment according to the measured reference signal received power (RSRP), the channel quality indicator and the time advance. Although these works demonstrated a high accuracy, the feature variation and correlation over time were not considered, which leaves room for improvement.

The temporal dimension of the problem was exploited in [20], by integrating DenseNet [32] with a stacked-LSTM module. The features used were the GPS signal, the ambient light and the magnetic intensity, the barometric pressure and the WiFi signal. Initially, each feature was processed through a 1D DenseNet, and consequently the various feature maps were concatenated and passed to a 3-layer LSTM followed by an MLP, which performed the final classification task. The DenseNet LSTM architecture yielded accurate results, however, it entailed a substantial computational complexity, requiring the estimation of up to approximately 280 million trainable parameters. More importantly, by choosing to associate a distinct DenseNet with each sensor feature, the cross-feature correlation was neglected during the transformation of the initial input sequence and the extraction of high-level features. Additionally, the proposed model was benchmarked only against threshold-based models, and there was no comparison with other ML-based or TSC models. Finally, the use of the GPS signal renders the proposed model inconvenient, since the majority of the cellular users have their GPS switched off and, more importantly, GPS is highly power consuming.

III. Problem Formulation and Proposed Method

A reliable IOD model ought to provide (i) a high IOD accuracy and (ii) a small environment change detection delay, between the actual and the predicted time at which an environment transition takes place. In addition, it is essential to ensure that there is good trade-off between these two measures and the complexity of the model. In what follows, we describe the process followed to structure our IOD problem, we briefly discuss existing DL models for multivariate TSC, and we introduce a new multivariate TSC DL model for IOD.

A. Data Collection and Problem Formulation

To study the performance of different DL multivariate TSC models, a set of eight different features was recorded over a 6-month period, using a HUAWEI P30 lite and two Redmi Note 9 Android smartphones. Specifically, the features captured are

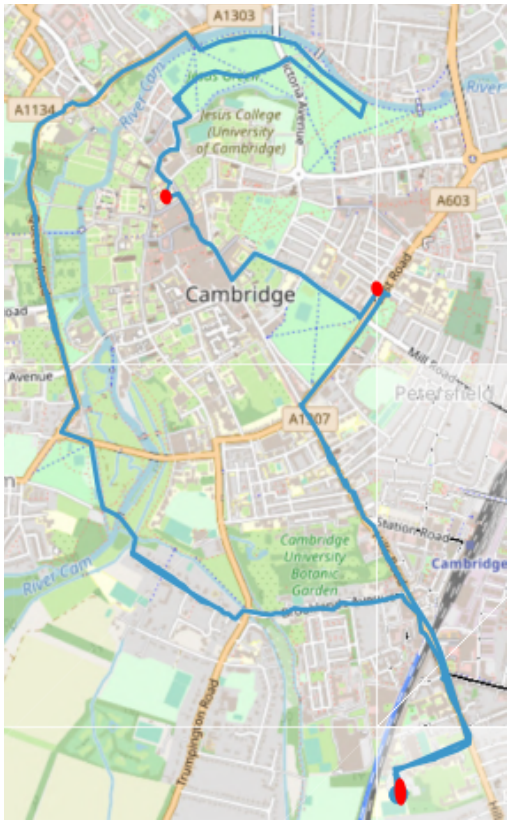


Fig. 1: Example of measurements collected in the city of Cambridge; the route is shown in blue, while the buildings visited are depicted as red dots.

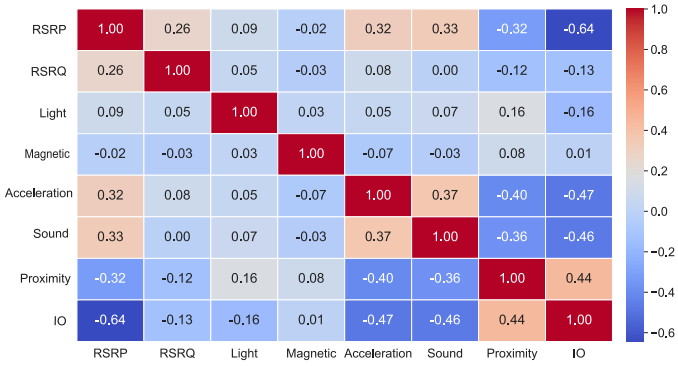


Fig. 2: Pearson correlation between the various sensor features and the environment type.

(i) the RSRP, (ii) the reference signal received quality (RSRQ), (iii) the total magnetic and (iv) the ambient light intensity, (v) the acceleration, (vi) and the sound intensity. Moreover, to assess the reliability of the ambient light sensor, we use the readings from (vii) the proximity sensor, indicating whether the light sensor is blocked by an obstacle, and (viii) we also include a binary label including whether the recording was taken during night or daytime. The sampling rate for all the sensors is set to 1 Hz and a min-max normalization is applied to the sensor data, that has been cleaned and had outliers removed. In particular, while inspecting the data we observed that substantial outliers existed at the top 1% of the samples distribution, presumably attributed to malfunction of the recording cellphone sensors. Hence, this data was removed before applying the min-max normalization.

The data set comprises approximately one million samples collected in three different countries, Greece, Luxembourg and the United Kingdom, at different sites including cities, countryside, villages and sub-urban areas. Moreover, the data depicts various aspects of the every-day life, such as work, entertainment (malls, cinemas, cafes, bars, etc.), traveling, physical exercise, etc. Furthermore, the recordings took place under various weather conditions such as light and heavy rain, clear sky, fog etc. An example of measurements taken in the city of Cambridge is shown in Fig. 1, where the path followed within the city is depicted in blue and the buildings visited are marked as red dots. Fig. 2 shows the Pearson correlation coefficient [33], ρ, which indicates the strength of linear association between the recorded features and the environment type. Hence, a ρ value close to 0 translates to a weak correlation between a sensor and user's environment, while values close to ± 1 suggest a strong correlation. A positive ρ implies that as the value of one feature increases, so does the value of the other feature, while the opposite holds for negative ρ values. For instance, as the strength of the cellular signal decreases, the IO label increases, i.e., takes values closer to 1, which implies an indoor environment.

To form a TSC problem, and create the time sequences to train and test the various DL models, a time window of length w is slid with a step s over the recorded data. Hence, each generated training or test sample $\mathbf{X}_i \in \mathbb{R}^{\mathrm{w} \times \mathrm{M}}$ is a multivariate time series comprising the values of M different features at w different time steps. Each sample $\mathbf{X}_i$ is associated to a binary target value, y_i, where 1 represents an indoor and 0 an outdoor environment. Hence, the objective of the multivariate TSC problem is to determine a user's environment at the current time step t, given the variation of the M different features within the time interval $(t - w, t]$.

B. Deep learning-based time series classification in a nutshell

In the context of DL, multivariate TSC is realized by applying consecutive non-linear transformations to the input time series. Assuming k classes, each time series sample, $\mathbf{X}_i$, is associated to a certain class to form an input-target pair, and in the output the DL model predicts the probability of $\mathbf{X}_i$ belonging to each class [34]. Eventually, the class with the highest probability is selected as the actual time series class. Indoor outdoor detection is a binary time series classification problem, where without loss of generality we assume that 1 corresponds to indoor and 0 to outdoor environments.

Convolutional neural networks (CNNs) and recurrent neural networks (RNNs), are among the common DL model types employed for TSC. The basic functionalities executed within CNNs are convolutional and local or global pooling operations. The convolutional operation is equivalent to applying a sliding window filter to the input time sequence, where the filter weights are estimated such as to minimize a loss function. Since a filter with a given set of weights can identify a specific pattern within the input sequence, by using multiple filters it

is possible to retrieve various patterns within the same input sequence. Let $w_f \times m_f$ be the filter dimensions, where w_f is the filter length and m_f is the number of features of the input sequence. Separate filter weights are estimated for every input feature sequence, and the elements of the final output sequence are computed through a dot product between the filter weights and the input sequence elements, followed by a cross-channel summation. According to the number of the filters, n_f, applied to the input sequence, the output of the convolutional layer, known as the feature map, can be a univariate (only one filter is used, $n_f = 1$) or a multivariate time series (where the number of output features is equal to n_f). Finally, pooling operations are used to reduce the size of the feature maps and aggregate information.

A convolutional neural network is a feedforward type neural network, i.e., the information is directed only from the input layer towards the output layer. Unlike CNNs, in RNNs the neurons of each layer are allowed to establish connections with themselves or with neurons found in previous layers. Hence, a neuron output does not depend only on the current input, but on the previous data history as well, rendering RNNs a reliable sequence modeling tool. The standard RNN model includes a self-loop at the hidden unit between the input and the output layer. Thus, the current hidden state of the RNN, h_t, is a non-linear function of the previous time step hidden state, h_{t-1}, and the current input, x_t:

$$h_t = f(x_t, h_{t-1}). \tag{1}$$

The main drawbacks of the standard RNN model is that it suffers from (i) the vanishing gradient problem, which can prevent the network from updating, and from (ii) the exploding gradient problem which can compromise long-term memory predictions, leading to severe accuracy deterioration. To eliminate the vanishing gradient problem, and mitigate the exploding gradient problem, LSTM networks can be leveraged. In addition to the hidden state, h, which carries short-term past information, LSTMs also have a cell state, c, which transfers long-term memory information. As shown in the orange box in Fig. 3a, the operation of the LSTM cell is regulated by three gates: the forget, the update and the output gate. The σ and the tanh notation in Fig. 3a denote the application of a sigmoid and a tanh activation function, respectively. The forget gate is responsible for deciding which part of the cell state will be kept and which part will be forgotten/zeroed. The update gate defines which cell state values will be updated and, finally, the output gate determines the hidden state values that will be forwarded to the next LSTM cell.

Deep LSTM architectures consist of multiple stacked LSTM layers, where for two consecutive layers, the hidden states of the lower layer are used as the input for the upper layer. To further improve the performance of such architectures, the use of attention mechanisms has been proposed [27], [35]. In that case, as shown in Fig. 3a, instead of receiving as an input the hidden states of the lower layer, the LSTM cells of the upper layer are fed with a context vector, $\widetilde{c}$, which conveys information about the entire input sequence. Additionally, in cases where there is a dependence between two consecutive cell outputs, e.g., in text translation tasks, the output of the previous time step can be provided as an extra input for the current LSTM cell. Thus, the current hidden state of the upper LSTM layer, h'_t, is now a non-linear function of the previous cell hidden state, h'_{t-1}, the previous cell output, y_{t-1}, and the current step context vector $\widetilde{c}_t$:

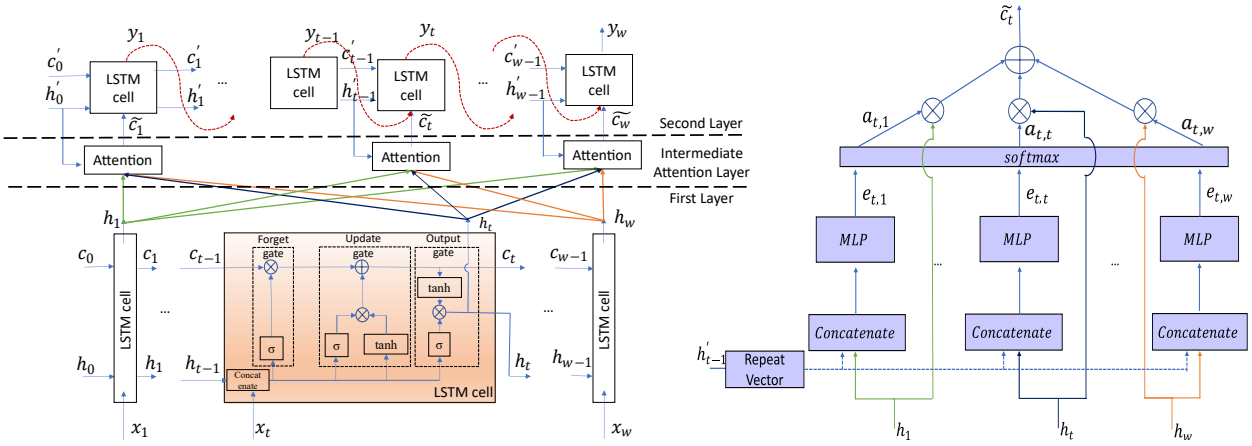
$$h'_t = f(y_{t-1}, h'_{t-1}, \widetilde{c}_t) \tag{2}$$

where y_{t-1} is estimated by applying a softmax activation function to h'_{t-1}, and the context vector is estimated as a weighted sum of the lower layer hidden states [27]:

$$\widetilde{c}_t = \sum_{j=1}^{w} a_{i,j} h_j \tag{3}$$

with the weights $a_{i,j}$ being computed by applying a softmax function over the respective alignment score $e_{i,j}$ [27]:

$$e_{i,j} = g(h'_{t-1}, h_j) \tag{4}$$



(a) LSTM cell structure and self-attention mechanism between two LSTM layers. (b) Intermediate self-attention layer structure.

Fig. 3: Block 3 of the proposed model.

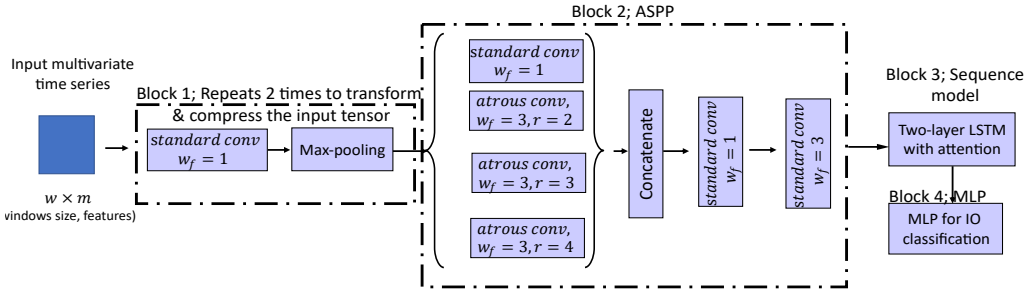


Fig. 4: Proposed model architecture.

where the alignment model g is parametrized via an MLP, as proposed in [27]. Figure 3b depicts the aforementioned procedure, showing the overall structure of a self-attention layer inserted between two LSTM layers.

C. Proposed Deep Learning Framework

In this subsection we present a new DL framework for multivariate TSC. As shown in Fig. 4, the proposed model consists of four different blocks: (i) a standard convolution with $w_f = 3$ followed by a max-pooling operation, (ii) an atrous spatial pyramid pooling (ASPP) [28] block, (iii) a two-layer LSTM module using self-attention [27], and (iv) an MLP used to carry out the final classification. From now on, we refer to this model as convolutional atrous spatial pyramid pooling attention LSTM (CAP-ALSTM). The motivation behind the use of the first two blocks prior to the LSTM with attention module, is that instead of processing raw sensor data, the sequence model receives at its input high level feature maps, which convey correlations between different features at different time steps. That is to say that the initial time series is transformed into a new one that will be easier for the sequence model to process. More importantly, in the case of a large window sizes, the initial sequence is compressed, effectively reducing the high computational cost during training time, associated with the complicated LSTM structure.

Specifically, the first block is used to identify correlations between different features of the input tensor at different time steps, and to compress useful information. The block is typically repeated 2 times and depending on the window size, w, the max-pooling operation can be skipped. The initial number of filters used in the convolutional layer is 32, and it is increased by a factor of two each time the block is repeated. The purpose of max-pooling is to ensure that the size of the output feature map is reduced before it is eventually forwarded to the stacked LSTM module, allowing the network to process larger time-sequences without comprising its computational efficiency. The size of the max-pooling layer is 2, i.e., each max-pooling operation halves the time sequence length, and in cases where w is already small we choose to omit the pooling layer.

The second block was introduced in [28] for image segmentation. However, in this paper we exploit its potential to capture multi-scale correlations for a TSC problem. In ASPP, multiple parallel atrous convolutional layers are employed to process the input feature map and unveil correlations in multiple scales between the feature map's elements. The output feature maps are then concatenated and passed to a standard convolutional layer with $w_f = 1$, used to fuse the information originating from the multiple atrous layers. The functionality of an atrous convolutional layer is similar to that of a standard layer, with the difference that the convolutional filter weights are dilated. This means that holes, i.e., zero elements, are placed between consecutive filter weights. The number of holes is equal to $r-1$, where r is a hyperparameter known as the dilation rate. The use of dilated filters with different dilation rates allows information to be captured originating from different temporal scales, and so identify correlations between distant points. As shown in Fig. 4, the ASPP employed comprises a standard convolution with $w_f = 3$, and three atrous convolutions with $w_f = 3$ and a dilation rate $r = 2, 3$, and 4, respectively. The number of filters used in these four convolutional layers is 64, while the standard convolutional layer used to fuse the concatenated feature map has 32 filters.

The feature map coming out of the ASPP is then processed by a stacked two-layer LSTM, where a self-attention mechanism is applied to the information forwarded from the lower towards the upper LSTM layer [27], as shown in Fig. 3. The attention mechanism employed between the two LSTM layers, allows the reconstruction of the time series, enabling the second layer to focus on the time series parts that are more relevant to the user environment prediction. Practically, this is an encoder-decoder architecture where the first layer encodes the input time series, and the second layer decodes it, pronouncing an environment type. Finally, the proposed network is terminated by an MLP with a 32 unit hidden layer and a single neuron output layer, used to complete the binary classification task.

D. Interpreting Machine Models with the SHAP framework

Machine learning models have been criticized for their lack of interpretability, since they are commonly considered as black boxes. Indeed, the consecutive non-linear operations applied to the input features make it hard to develop an intuition between the connection of the inputs and the output of an ML model. However, recently, a unified framework was introduced to shed light on the complicated operations that take place within an ML model [29]. The SHAP framework attempts to explain the response of an ML model through a set of linear polynomial functions with binary random variables. Let u be a polynomial explanation function, then [29]:

$$u(x') = \phi_o + \sum_{i=1}^{M} \phi_i x'_i \tag{5}$$

where $x' \in \{0,1\}^M$ are the binary random variables which are mapped to initial input space through a function v, $x = v(x')$, and M is the number of the input features. The coefficients of the explanation function must be selected such as $u(x') \approx f(v(x'))$, and for a specific input, x', they quantify the contribution of each feature towards the output. Moreover, the explanation functions u must satisfy a local accuracy, a missingness and a consistency property [29]. In [29], it was shown that there is a unique explanation function that satisfies these three requirements, for which the polynomial coefficients are equal to the Shapley values [36]:

$$\phi_i(f,x) = \sum_{z' \subset x'} \frac{|z'|!(M-|z'|-1)!}{M!} [f(z') - f(z' \backslash i)] \tag{6}$$

where $z' \subset x'$ corresponds to all the possible vectors z' whose non-zero entries are a subset of the entries of x'. The notation $|\cdot|$ denotes the number of non-zero entries in a vector, while the notation $z' \backslash i$ signifies setting $z' = 0$ for the i-th feature, and retaining the rest of the entries. The terms of the difference in the summation represent the response of the black box with and without the i-th feature, whose impact we seek to quantify. By subtracting the two responses, one can isolate the contribution attributed to feature i. The difference is also weighted according to the number of non-zero features included in z'. It should be noted that Shapley values consider all the possible permutation in x'. Hence, the SHAP framework does not account only for the sole impact of a feature, but it also considers how the feature interacts with the rest of the features to affect the model's output.

The Shapley values can be estimated using model-agnostic approximations, such as the Kernel SHAP, or the model-specific approximations, such as DeepSHAP [29]. A problem that arises in both cases, is that the calculation of the Shapley values in (6) entails removing the i-th feature from the input tensor. Since the input tensor of an ML model assumes a constant size, it is infeasible to remove entirely an input feature. Instead, the value of the feature that needs to be excluded can be replaced by a random value of the same feature coming from the training data set. By doing so for all the possible subsets z', the effect of the feature to be removed will be eventually sampled out. Consequently the coefficients are straightforwardly estimated through (6) with x being sampled from the test set.

IV. Results

In this section, we explore the performance of different IOD models. To that end, we use 6-fold cross-validation, evaluating the performance of each IOD model over six different subsets of our entire data set, each containing approximately $N \approx 800,000$ samples. As shown in Fig. 5, the first subset comprises the initial N samples of our data set, while the rest of the subsets are created by sliding the starting point of the subset by approximately $N_s \approx 150,000$ samples and

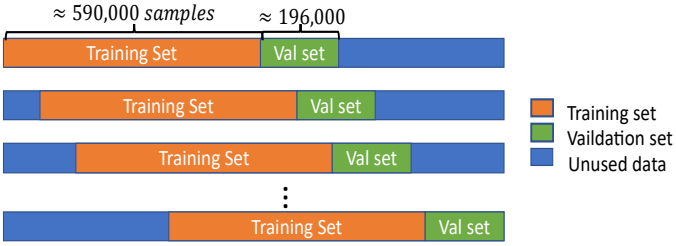


Fig. 5: Generation of the validation sub-sets.

considering the following N samples. To avoid information leakage from future towards past samples in the case of DL TSC models, all the models are trained using the first 590,000 samples of each subset, whilst their performance is measured with respect to the remaining samples. All the models are trained on a Nvidia Quadro RTX 8000 GPU over Tensorflow, using the Adam optimization algorithm for 250 epochs, with the initial learning rate set to 0.001, and a batch size equal to 512. Finally, the loss function sought to be optimized is the binary crossentropy loss function.

A. IOD models considered

In our work, we compare the performance of different IOD models: ML-based classification models, DL TSC models and threshold-based models. For the ML-based IOD models, the feature variation over time is neglected [15], [24], [25]. We consider two ML architectures that have been used in the past in the context of IOD [15], [17], [24]. Specifically, we train (i) an MLP consisting of 5 hidden layers with 1024, 512, 256, 128, and 64 neurons for each hidden layer, respectively, [24], [25] and (ii) a random forest (RF) classifier with 100 trees [15]. To avoid overffiting in the MLP a dropout layer, with a dropout rate equal to 0.2, is placed after each hidden layer.

Additionally, four DL TSC models are explored for the IOD classification problem: (iii) a DenseNet-LSTM [20], (iv) an LSTM (v) an LSTM with attention (ALSTM), and (vi) the CAP-ALSTM model. The DenseNet-LSTM used is similar to that of [20]. A distinct DenseNet is used to process separately each input feature and generate higher-level features. Hence, we use 5 parallel DenseNets receiving as an input the (i) cellular signal readings, (ii) the magnetic sensor readings, (iii) the light intensity, the daytime and the proximity indicator, (iv) the accelerometer readings, and (v) the sound sensor measurements, respectively. Each DenseNet comprises two dense blocks connected with an intermediate transition layer. The dense blocks use standard convolutions with $w_f = 3$, they assume a compression rate equal to 0.5, and they have a depth and a growth rate equal to 18 and 12, respectively. For more information regarding the dense block structure and functionality we refer the reader to [32]. The feature maps of the individual DenseNets are concatenated and forwarded to a three-layer LSTM, followed by an MLP. We preserve the same number of hidden state units per LSTM cell as in [20], i.e., 4096, 2048, and 1024. Finally, the MLP hidden layer has 1024 neurons connected to single output neuron which determines the environment type.

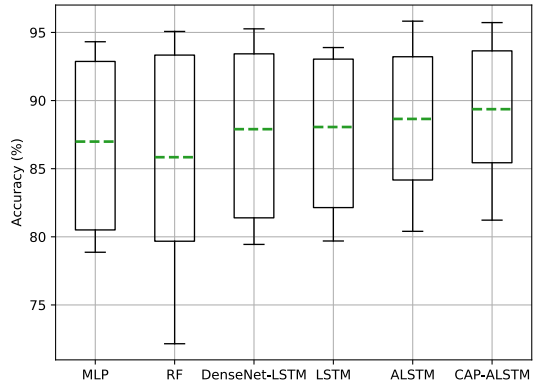
The LSTM and ALSTM models consist of two stacked layers followed by an MLP layer used to carry out the final classification. For the LSTM the first and the second layers have 64 and 32 hidden state units, respectively, while the MLP has 32 neurons. In the ALSTM case, both recurrent layers have 32 hidden state units and the attention mechanism described in Section III-B is employed to weight the information transferred from the lower to the upper LSTM layer. The ALSTM practically corresponds to Blocks 3 & 4 of Fig. 4, i.e., the CAP-ALSTM without the convolutional layers. The parameters for the convolutional layers of the CAP-ALSTM model are shown in Fig. 4. Blocks 3 & 4 share the same parameters as the ALSTM.

Finally, we also consider the performance of two threshold-based models, the IODetector and the SenseIO [14], [16]. As discussed in Section II, for IODetector the light, the magnetic and the cellular signal readings are processed separately by a detector, which provides an environment type along with a confidence level associated with its decision. For each detector we use the same thresholds as those in [14]. For the cellular detector, whose decision is made base on the signal strength difference, we only considered the RSRP variation of the serving base station. For SenseIO, we only use the recordings of the light and the cellular sensor, since our data set does not include WiFi readings. Additionally, we do not incorporate the activity recognition module, since it was implemented through an external application. The thresholds used are the same as those in [16].

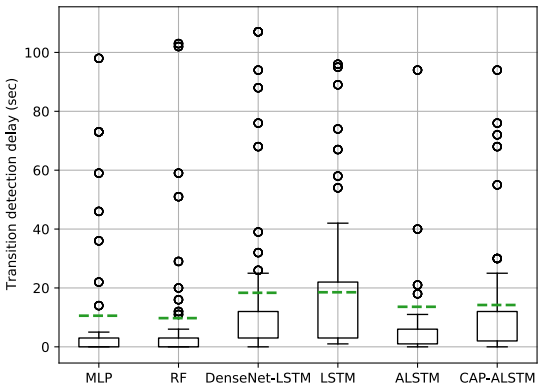
To evaluate the performance of each model we inspect their accuracy and their F_1-score, in order to account for unbalances between the two classes. Additionally, we provide statistics of the delay (mean, median and standard deviation), d, when there is an environment transition. To explore the computational efficiency of each model we record the number of its parameters, its memory requirements, the simulation time required per prediction, and the training time.

B. Comparison between different models

The classification results for $w = 6$ and $s = 1$ are presented in Table I, while Figs. 6a and 6b show the accuracy and the delay boxplot, respectively, over the 6-sub sets for each model. The best performances in Table I are shown in bold black color, while the worst with dark red. As we can observe, in terms of accuracy, all the DL TSC models perform better than the standard classifiers, which suggests that exploiting the temporal dimension is beneficial for an IOD model. Among the models considered, the CAP-ALSTM demonstrates the best performance compared to the standard classifiers, having approximately a 2% better accuracy. More importantly, as we can see in Fig. 6a, the CAP-ALSTM yields the best upper and lower accuracy bounds over the 6 validation sets. On the other hand, the standard classifiers exhibit the worst lower bound, with the RF accuracy deteriorating to approximately 72% in the worst case scenario. Additionally, it can be observed that the use of the attention mechanism between the LSTM layers can increase the accuracy of the sequence model, and lead to improved lower and upper quantiles. All the models yield a high F_1 score, indicating that both classes are treated fairly. However, we note that for the given window value, the difference in the accuracy and the F_1 score between the various DL TSC models is relatively small. We can



(a) Accuracy boxplot for the ML-based and DL TSC IOD models.



(b) Delay boxplot for the ML-based and DL TSC IOD models.

Fig. 6: Accuracy and delay boxplot for the ML-based and DL TSC IOD models. The green dashed line represents the mean value.

TABLE I: IOD TEST SET PERFORMANCE OF THE ML-BASED AND DL TSC MODELS.

	(i) MLP [24], [25]	(ii) RF [15], [25]	(iii) Dense-LSTM [32]	(iv) LSTM	(v) ALSTM	(vi) CAP-ALSTM
Accuracy (%)	86.98 ± 6.50	85.59 ± 8.42	88.05 ± 6.42	88.06 ± 6.05	88.65 ± 5.69	**89.36 ± 5.28**
F_1-score	89.14 ± 5.80	87.75 ± 7.60	89.84 ± 5.56	90.23 ± 5.55	90.33 ± 5.44	**90.97 ± 5.06**
d (sec)	10.57 ± 23.50	**9.63 ± 23.65**	18.35 ± 29.71	18.53 ± 27.26	13.60 ± 16.51	14.20 ± 23.12
Median d (sec)	2	**2**	4.00	5.00	3.00	3.00
Training Time (sec)	312	**123**	68440	2220	3650	4458
Single Prediction Time (msec)	6.3	7.1	17.1	**3.7**	4.1	4.9
Required Memory (Mb)	16.4	**2.9**	500.1	4.5	3.5	5.4
Model Parameters	176K	-	146M	32K	**17K**	74K

TABLE II: COMPARISON BETWEEN THE CAP-ALSTM AND TWO THRESHOLD-BASED MODELS.

	CAP-ALSTM	IODetector [14]	SenseIO [16]
Accuracy	**89.36 ± 5.28**	68.10 ± 8.47	67.2 ± 5.80
F_1-score	**90.97 ± 5.06**	77.7 ± 6.79	77.8 ± 5.00
d (sec)	14.20 ± 23.12	**11.6 ± 11.85**	19.51 ± 23.37
Median d (sec)	**5**	9	**5**

also observe that preprocessing the input time series, i.e., models (iii) and (vi), can lead to a higher accuracy than using raw sensor data (LSTM and ALSTM). For instance, CAP-ALSTM demonstrates a better lower quartile than ALSTM, while DenseNet-LSTM exhibits a better higher top quartile than LSTM.

As shown in Table I and Fig 6b, the environment transitions are detected by all models on average in less than half a minute, which can be definitely considered as a tolerable delay. We can observe that the standard classifiers demonstrate a smaller average delay than the DL TSC models using LSTMs. However, the use of the attention mechanism can reduce the average delay of the LSTM by approximately 5 seconds. From Fig. 6b, we can observe that there are outliers in the delay distribution, thus the models yield a mean and a standard deviation much greater than the median. This suggests that some environment transitions can be detected easier than others for all the models.

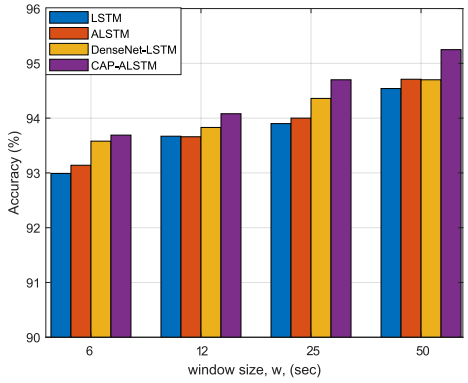
Finally, a comparison between the CAP-ALSTM and the two threshold-based models is shown in Table II. Both threshold-based models yield an accuracy close to 70%, which is approximately 20% lower than that of CAP-LSTM. Similar accuracy levels have also been reported in [15] and [20]. Part of the significantly deteriorated accuracy could be due to the fact that we did not consider the full threshold model implementation. As mentioned, for SenseIO we do not include the WiFi detector component, while for the IODetecor we only consider the cellular signal variation of the serving base station. We can also observe that the average transition delay for IODetector is smaller. This can be attributed to the cellular detector component, which can effectively capture abrupt RSRP variations when the environment changes.

Regarding the computational efficiency of the DL models, the DenseNet-LSTM architecture appears to be the most computationally demanding, as it entails three orders of magnitude more parameters than the other models. Indeed, the model proposed in [20] associates an individual DenseNet with every input feature. The DenseNet architecture encompass cascaded convolutional layers with direct connections emerging from every layer to all subsequent layers, which results in an increased computational cost. On top of that, by choosing to have a separate DenseNet for each feature, the cross-feature correlation is neglected during the preprocessing of the initial input time series. Furthermore, the authors in [20] used a tremendous number of LSTM hidden units, which appears to be redundant as the same levels of accuracy can be achieved with substantially more lightweight models.

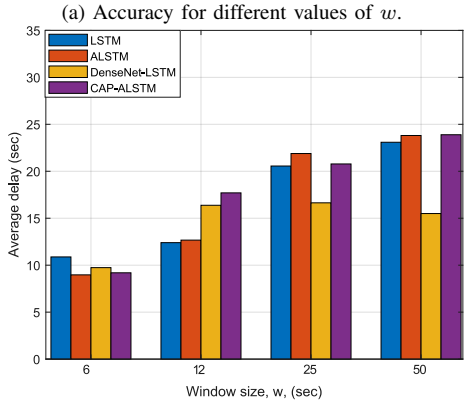
C. Impact of window size

Consequently, we evaluate the performance of each model as the window size w increases, i.e., as we take into account a larger number of the past sensor readings. Four different windows sizes are considered: 6, 12, 25, and 50 seconds. In the implementation of the proposed model, we skip the max-pooling layer for $w = 6$ and 12, whilst for $w = 25$ and 50 we use 1 and 2 max-pooling layers, respectively. In this manner, we can reduce the training computational cost by preserving sequences with small length in the input of the ALSTM. Specifically, the input sequence passed to the ALSTM has a length equal to 12, for $w = 25$ and 50. For the DenseNet-LSTM model, we change the number of the LSTM hidden state units, since we find that those used in [20] are unnecessarily large, resulting in computationally exhaustive requirements and overextended training times. Specifically, in what follows, the concatenated vector from the DenseNets is fed to a two-layer LSTM, instead of a three-layer, with 64 and 32 hidden state units, respectively. The parameters for the rest of the models remain the same.

Figures 7a and 7b show the accuracy and the average delay of each model, respectively. In terms of accuracy, we can see that proposed model outperforms the other DL TSC models as the window size increases. In particular, the accuracy is increased from 93.5% up to approximately 95.3%, which is almost a 2% accuracy increase. That is attributed to the use of the ASPP block and the attention mechanism, which



(a) Accuracy for different values of w.



(b) Average environment transition detection delay for different values of w.

Fig. 7: Accuracy and average delay of the DL TSC models under consideration, for different values of w.

becomes more effective for longer time sequences. When the dilation rate is comparable to the window size, the number of valid filter weights is small (i.e., weights multiplied with valid, non zero-padded, input feature map regions). Thus, as w increases the convolutional layers of the ASPP with larger dilation rate become more effective (i.e., applied in larger non-zero areas), and they are capable of capturing multi-scale correlation within the input sequence. For $w = 12, 25, 50$, the feature map forwarded to ASPP the has a length equal to 12 due to the max-pooling, however as w increases the feature map conveys more purposeful information, and hence the multi-scale sampling becomes more efficient. We can also observe that a simple attention mechanism enables the LSTM to achieve the same accuracy level as the Dense-Net LSTM, but with a considerably smaller number of model parameters.

In terms of delay, a common trend identified for all the models is that a larger window size entails a larger average delay between the actual and the predicted transition moment. Again, an environment transition is detected by all models on average in less than half a minute. Initially, for $w = 6$ the ALSTM and the CAP-ALSTM demonstrate the smallest delay. However, as the window size is increased, we can observe that for $w = 25$ and 50 the delay for the DenseNet-LSTM remains constant at around 15 seconds, thus yielding the smallest delay. As a final comment, we note that increasing the window size does not severely affect the computational efficiency of our model. Indeed, the time required to train our model was only 5 hours compared to approximately 11 and 28 hours required for the LSTM and ASLTM, respectively. That is due to the downsampling of the initial input sequence, which allows the sequence length to be decreased, avoiding an overextended backpropagation through time during the training phase. Therefore, it is evident that the proposed model can meet the high accuracy and small delay objectives better than the other DL TSC models.

V. Deep Learning Model Interpretation and Energy Consumption

In this section we discuss the power consumption of the proposed model and we also investigate which are the features that assume the most pronounced role in its decision making.

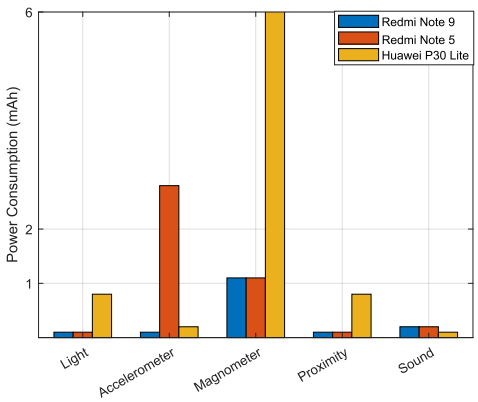


Fig. 8: Power consumption of each sensor type for the cellphones used to collect our data.

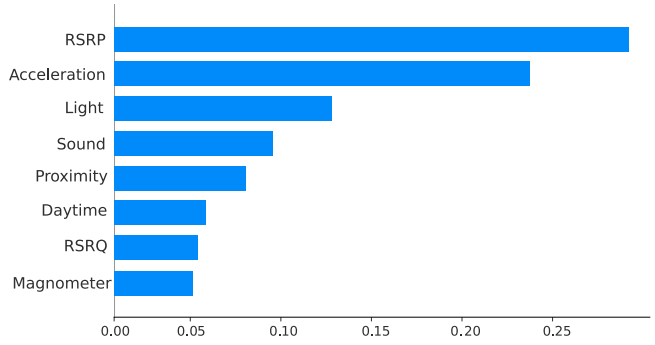


Fig. 9: Shapley values for the CAP-ALSTM; average contribution of each feature towards the characterization of a user's environment.

The power required by each sensor at each cellular phone, as recorded by the application used to collect our data, is shown in Fig. 8. The consumption of the cellular signal is not presented, as the associated data are part of the normal operation of the mobile device, and did not entail the recording of additional signals. The accumulated power consumption ranges from 2.6 mAh up to 8.7 mAh for the 3 cell phone models. The power consumption of the GPS or WiFi sensors, which have also been employed for IOD [20], [22], [23], is typically much higher than that of the sensors used in our work. For instance, in [15] it was reported that the GPS and the WiFI sensors can consume up to 6 times more power than the sensors used in our work.

An important observation from Fig. 8 is that for the same sensors the power consumption could vary from cell phone to cell phone. For instance, one can observe that for the Huawei P30 Lite the power consumption of the magnetic field sensor is substantially higher than that of the other models. Hence, it is necessary to develop an intuition behind the contribution of each sensor to the final model's decision. In the event that the contribution is negligible, while the power consumption is high, one could simply ignore this feature.

To achieve this, we leverage the SHAP framework describe in Section III-D. The Shapley values for the pretrained CAP-ALSTM model are estimated through (6), where f depicts all the consecutive non-linear operations of the CAP-ALSTM applied to the input time series. The contribution of each feature is estimated using the Kernel SHAP [29], and the Shapley values for the CAP-ALSTM are shown in Fig. 9. The three most important features are the RSRP, the linear acceleration and the light intensity, while the three less important features are the daytime, the RSRQ and the magnetic field. Although recording the RSRQ does not pose additional power requirements, from Fig. 8 we can observe that the magnetometer consumes the highest power for Huawei P30 Lite and the second highest power for the other two cellphones. Hence, we consequently explore what is the performance of our model when trained without these two features. In that case, the average accuracy for the CAP-ALSTM over the six validation sets of Section IV-B is 88.83% compared to the initial 89.36% when using all the features, i.e., we have a 0.5 % reduction but it still remains higher than that of the rest of the models. The mean transition detection delay is

increased by approximately 1.5 seconds. However, the power consumption is substantially reduced for all the cellphones, with the accumulated power consumption ranging now from 1.5 mAh to 3.2 mAh, without jeopardizing the effectiveness of the DL TSC model.

VI. Conclusion

In this paper we evaluated the performance of various DL TSC models for IOD. Efficient IOD models are of high significance for location-based services and seamless navigation systems, which are an inextricable component of the IoT ecosystem. Unlike previous work in the field, we leveraged the sensor reading variation over time to enhance the environment detection accuracy and ensure a small environment transition delay. We introduced a new DL TSC model, building on the concept of self-attention and ASPP. We showed that a multivariate TSC perspective of the IOD problem can be advantageous both in terms of IOD accuracy and transition detection delay. The proposed DL TSC model outperforms the other models studied, especially for longer time series, yielding the highest accuracy, a small delay for the environment transition detection, and having significantly smaller computational requirements and using only low-power consumption sensor readings.

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (IoT): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, 2014.

[3] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 414–454, 2013.

[4] O. B. Sezer, E. Dogdu, and A. M. Ozbayoglu, "Context-aware computing, learning, and big data in internet of things: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 1–27, 2017.

[5] L. Chen, S. Thombre, K. Järvinen, E. S. Lohan, A. Alén-Savikko, H. Leppäkoski, M. Z. H. Bhuiyan, S. Bu-Pasha, G. N. Ferrara, S. Honkala, *et al.*, "Robustness, security and privacy in location-based services for future IoT: A survey," *IEEE Access*, vol. 5, pp. 8956–8977, 2017.

[6] H. Huang, G. Gartner, J. M. Krisp, M. Raubal, and N. Van de Weghe, "Location based services: Ongoing evolution and research agenda," *Journal of Location Based Services*, vol. 12, no. 2, pp. 63–93, 2018.

[7] C. Monn, "Exposure assessment of air pollutants: A review on spatial heterogeneity and indoor/outdoor/personal exposure to suspended particulate matter, nitrogen dioxide and ozone," *Atmospheric environment*, vol. 35, no. 1, pp. 1–32, 2001.

[8] B. Ye, K. Liu, S. Cao, P. Sankaridurg, W. Li, M. Luan, B. Zhang, J. Zhu, H. Zou, X. Xu, *et al.*, "Discrimination of indoor versus outdoor environmental state with machine learning algorithms in myopia observational studies," *Journal of translational medicine*, vol. 17, no. 1, p. 314, 2019.

[9] H. Jia, Y. Zhang, and W. Kong, "Indoor/outdoor detection for seamless positioning," *Sensors & Transducers*, vol. 171, no. 5, p. 283, 2014.

[10] J. Cheng, L. Yang, Y. Li, and W. Zhang, "Seamless outdoor/indoor navigation with WiFi/GPS aided low cost inertial navigation system," *Physical Communication*, vol. 13, pp. 31–43, 2014.

[11] Q. Zeng, J. Wang, Q. Meng, X. Zhang, and S. Zeng, "Seamless pedestrian navigation methodology optimized for indoor/outdoor detection," *IEEE Sensors J.*, vol. 18, no. 1, pp. 363–374, 2017.

[12] 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.

[13] L. Ravindranath, C. Newport, H. Balakrishnan, and S. Madden, "Improving wireless network performance using sensor hints," in *Proc. USENIX NSDI*, vol. 11, 2011.

[14] 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.

[15] V. Radu, P. Katsikouli, R. Sarkar, and M. K. Marina, "A semi-supervised learning approach for robust indoor-outdoor detection with smartphones," in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, 2014, pp. 280–294.

[16] M. Ali, T. ElBatt, and M. Youssef, "SenseIO: Realistic ubiquitous indoor outdoor detection system using smartphones," *IEEE Sensors J.*, vol. 18, no. 9, pp. 3684–3693, 2018.

[17] 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.

[18] I. Saffar, M. L. A. Morel, K. D. Singh, and C. Viho, "Machine learning with partially labeled data for indoor outdoor detection," in *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, IEEE, 2019, pp. 1–8.

[19] S. Li, Z. Qin, H. Song, C. Si, B. Sun, X. Yang, and R. Zhang, "A lightweight and aggregated system for indoor/outdoor detection using smart devices," *Future Generation Computer Systems*, vol. 107, pp. 988–997, 2020.

[20] Y. Zhu, H. Luo, F. Zhao, and R. Chen, "Indoor/outdoor switching detection using multi-sensor densenet and LSTM," *IEEE Internet Things J.*, 2020.

[21] I. Ashraf, S. Hur, and Y. Park, "MagIO: Magnetic field strength based indoor-outdoor detection with a commercial smartphone," *Micromachines*, vol. 9, no. 10, p. 534, 2018.

[22] G. Shtar, B. Shapira, and L. Rokach, "Clustering Wi-Fi fingerprints for indoor–outdoor detection," *Wireless Networks*, vol. 25, no. 3, pp. 1341–1359, 2019.

[23] Y. Zhu, H. Luo, Q. Wang, F. Zhao, B. Ning, Q. Ke, and C. Zhang, "A fast indoor/outdoor transition detection algorithm based on machine learning," *Sensors*, vol. 19, no. 4, p. 786, 2019.

[24] L. Wang, L. Sommer, T. Riedel, M. Beigl, Y. Zhou, and Y. Huang, "NeuralIO: Indoor outdoor detection via multimodal sensor data fusion on smartphones," in *International Summit Smart City 360°*, Springer, 2019, pp. 127–138.

[25] W. Wang, Q. Chang, Q. Li, Z. Shi, and W. Chen, "Indoor-outdoor detection using a smart phone sensor," *Sensors*, vol. 16, no. 10, p. 1563, 2016.

[26] T.-H. Yi, H.-N. Li, and M. Gu, "Effect of different construction materials on propagation of GPS monitoring signals," *Measurement*, vol. 45, no. 5, pp. 1126–1139, 2012.

[27] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[28] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected

crfs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, 2017.

[29] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st international conference on neural information processing systems*, 2017, pp. 4768–4777.

[30] S. Bakirtzis, "Research data supporting "Deep learning-based multivariate time series classification for indoor outdoor detection"," 2022. DOI: 10.17863/CAM.82668. [Online]. Available: https://doi.org/10.17863/CAM.82668.

[31] R. Janaswamy, *Radiowave propagation and smart antennas for wireless communications*. Springer Science & Business Media, 2001.

[32] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, "Densenet: Implementing efficient convnet descriptor pyramids," *arXiv preprint arXiv:1404.1869*, 2014.

[33] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise reduction in speech processing*, Springer, 2009, pp. 1–4.

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

[35] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[36] L. S. Shapley, H. Kuhn, and A. Tucker, "Contributions to the theory of games," *Annals of Mathematics studies*, vol. 28, no. 2, pp. 307–317, 1953.