



# Learning-powered migration of social digital twins at the network edge

Olga Chukhno <sup>a,b</sup>, Nadezhda Chukhno <sup>c</sup>, Giuseppe Araniti <sup>a,b</sup>, Claudia Campolo <sup>a,b,\*</sup>,  
Antonio Iera <sup>d,b</sup>, Antonella Molinaro <sup>a,b</sup>

<sup>a</sup> DIIES Department, University Mediterranea of Reggio Calabria, 89124 Reggio Calabria, Italy

<sup>b</sup> Consorzio Nazionale Interuniversitario per le Telecomunicazioni (CNIT), 43124 Parma, Italy

<sup>c</sup> ICT Faculty, Tampere University, 33014 Tampere, Finland

<sup>d</sup> DIMES, University of Calabria, 87036 Arcavacata di Rende, Italy

## ARTICLE INFO

### Keywords:

Social Internet of Things  
Edge computing  
Digital twin  
Migration  
Machine learning

## ABSTRACT

Digital Twins (DTs), which are paired to Internet of Things (IoT) devices to represent them and augment their capabilities, are gaining ground as a promising technology to enable a wide variety of applications in the sixth-generation (6G) ecosystem, ranging from autonomous driving to extended reality and metaverse. In particular, “social” IoT (SIoT) devices, which are devices capable to establish social relationships with other devices, can be coupled with their virtual counterparts, i.e., social DTS (SDTs), to improve service discovery enabled by browsing the social network of friend devices. However, the mobility of SIoT devices (e.g., smartphones, wearables, vehicular on board units, etc.) may require frequent changes in the corresponding SDT placement in the edge domain to maintain a low latency between the physical device and its digital replica. Triggering SDT relocation at the right time is a critical task, because an incorrect choice could lead to either increased delays or a waste of network resources. This work proposes a learning-powered social-aware orchestration that predicts the mobility of SIoT devices to make more judicious migration decisions and efficiently move the paired SDTs accordingly, while ensuring the minimization of both intra-twin and inter-twin communication latencies. Different machine learning (ML) and deep learning (DL) algorithms are used for SIoT device mobility prediction and compared in terms of a wide set of meaningful metrics in order to identify the model that achieves the best trade-off between prediction accuracy and inference times under different scenarios. Simulation results showcase the improvements of the proposal in terms of reduced network overhead (by up to a factor of 3) and intra-twin and inter-twin communication latency (by up to 10%) compared to a more traditional solution, which activates the relocation of the DTs at fixed time intervals following periodic optimizations.

## 1. Introduction

Fifth-generation (5G) networks are aimed at providing three types of services as identified by the International Telecommunication Union (ITU), which are [1]: massive machine-type communication (mMTC), ultra-reliable low-latency communication (uRLLC), and enhanced mobile broadband (eMBB). Among them, mMTC caters to a vast number of typically resource-constrained heterogeneous Internet of Things (IoT) devices, establishing data communications without human interaction and for which high data rate support is not critical.

According to the Social Internet of Things (SIoT) paradigm [2], IoT devices are enabled to establish friendship relationships as humans do. “Friend” devices can be located in the same place, carried by people who meet frequently, or belong to the same model, vendor, or production batch. Establishing social relationships (e.g., with devices

of the same brand, of the same owner, in the same place) enables IoT devices to effectively search through social links for the desired services/data provided by their friend trustworthy devices [3]. For instance, a device may ask for software updates to devices of the same brand or for location-based services to devices in the same area.

In recent years, the digital twin (DT) concept has attracted significant interest in the IoT domain [4] and is expected to become a key pillar for more demanding sixth-generation (6G) IoT deployments [5]. Indeed, for a wide range of disruptive applications, from self-driving cars navigating complex environments to immersive experiences in metaverse environments, there is a need for technological advancements and evolutions towards 6G networks, providing higher bandwidth, higher connection density, lower latency, among others, compared to 5G systems [6].

\* Corresponding author at: DIIES Department, University Mediterranea of Reggio Calabria, 89124 Reggio Calabria, Italy.

E-mail addresses: [olga.chukhno@unirc.it](mailto:olga.chukhno@unirc.it) (O. Chukhno), [nadezhda.chukhno@tuni.fi](mailto:nadezhda.chukhno@tuni.fi) (N. Chukhno), [araniti@unirc.it](mailto:araniti@unirc.it) (G. Araniti), [claudia.campolo@unirc.it](mailto:claudia.campolo@unirc.it) (C. Campolo), [antonio.iera@dimes.unical.it](mailto:antonio.iera@dimes.unical.it) (A. Iera), [antonella.molinaro@unirc.it](mailto:antonella.molinaro@unirc.it) (A. Molinaro).

<https://doi.org/10.1016/j.comcom.2024.07.019>

Received 6 February 2024; Received in revised form 30 June 2024; Accepted 30 July 2024

Available online 7 August 2024

0140-3664/© 2024 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

As digital counterparts of physical entities, DTs mimic the properties and conditions of real-life objects through models and data [7], while also predicting their behaviors. Furthermore, DTs can virtually augment the capabilities of IoT devices with limited computing, storage, and battery resources.

In particular, DTs endowed by social component, i.e., social DTs (SDTs), can improve data exchange and facilitate service discovery across the SIoT network [8–10] by leveraging information exposed by SDTs on behalf of resource-constrained physical devices. SDTs can be placed at the network edge to guarantee near real-time interactions with their physical counterparts, the so-called *intra-twin communications* [11]. In addition, applications that may need to quickly browse the social network by visiting SDTs of “friend” devices, e.g., to search for services by asking trusted entities or to build cooperative ones, could benefit from looking for those SDTs in the same edge server or closer ones, to experience low-latency *inter-twin* communications.

One challenge in this context is the mobility of SIoT devices [12]. SIoT devices may be very heterogeneous, ranging from smartphones to wearables carried by people [13] to vehicular on-board units installed in cars [14]. This diversity results in different mobility profiles, from pedestrian to vehicular patterns. A person’s dynamic and frequently changing movement profile is characterized by stops, changes in direction, and potentially irregular speeds. In contrast, cars present a more predictable mobility profile featuring higher speeds, well-defined routes, and fewer stops.

Device movements may necessitate frequent changes to SDT placement, with SDTs that may need to migrate among edge servers whenever a physical device roams across cells and changes connectivity points, which may be more or less frequent depending on the specific application, such as in the case of vehicles, drones, or pedestrian motion. Problems related to SDT migration increase when the cell coverage decreases, i.e., when the connection between physical and digital devices operates at extremely high frequencies, such as millimeter-wave or terahertz, and/or to serve high-density networks [15].

Although the general issue of DT migration is quite well investigated in the recent literature [16–19], it becomes more challenging for SDTs. This is because the migration of an SDT may imply that the SDTs of its friend devices are to be migrated as well to meet the targeted overall navigability latency in the SIoT network. The frequency at which the SDT relocation decisions are triggered is a crucial factor to consider. Frequently triggering relocation can incur high migration costs in terms of the amount of SDT-related data transfer between edge servers and of increased complexity. However, rarely triggering relocation may severely decrease the system performance.

Triggering the relocation of SDTs either at fixed intervals, regardless of the device mobility, as we proposed in our previous work [8], or simply according to the *current position* of SIoT devices, as in [20], results in suboptimal performance and does not adequately capture the device mobility and requirements.

In this context, the present work builds on our previous proposal [8] by providing the following main original contributions:

- We define the SDT interfaces, functions, and components and augment SDTs with machine learning (ML) and deep learning (DL) capabilities fed by the data retrieved from the paired physical devices to predict their mobility.
- We propose a novel learning-powered SDT migration algorithm, where the decision about whether a relocation of an SDT needs to be triggered is taken according to (i) the predicted future mobility of the SIoT devices and (ii) latency requirements.
- A set of popular ML and DL algorithms, i.e., Support Vector Regression (SVR), ensemble regression learning methods such as Random Forest (RF) and Extreme Gradient Boosting (XGBoost),  $k$ -Nearest Neighbors ( $k$ NN), and Neural Networks (NNs) such as Long Short-Term Memory (LSTM) network, are compared among each other and evaluated in different scenarios (datasets) and

also through a cross-validation approach to identify the most suitable model for mobility prediction to be implemented in the SDT ecosystem.

- An extensive performance evaluation campaign is conducted under relevant scenarios to show the advantages of the proposal both compared to a solution that blindly triggers the repositioning of SDTs between edge servers at pre-established time intervals and to the ground truth in which the movement of the device is *a priori* known.

The remainder of this paper is organized as follows. In Section 2, we present the motivation for this research and review the related works. In Section 3, we introduce a learning-powered social-aware orchestration that predicts the mobility of devices to make better migration decisions and efficiently move SDTs accordingly while minimizing both intra-twin and inter-twin communication latencies. The simulation results are presented in Section 4. Finally, in Section 5, conclusions are drawn, and future works are outlined.

## 2. Background and motivations

### 2.1. IoT device mobility

SIoT devices are highly heterogeneous in computing, storage, and connectivity capabilities, and they differ in mobility patterns [12]. They encompass smartphones, wearables, devices on board the vehicle, sensors and actuators, printers, laptops, tablets, smart glasses, etc. Moreover, they may be carried by humans during their daily activities across different locations, or they can be statically placed.

Understanding, characterizing, and predicting device mobility have been the subject of numerous research experiments and studies in various fields outside communication engineering, such as psychology, neuroscience, and intelligent transportation. Indeed, device mobility is affected by several dimensions, as discussed in the following.

**Time-dependent mobility.** One phenomenon that affects device mobility is time, and recent literature has broadly examined daily patterns. In [21], the busiest days of the week and hours have been analyzed in terms of device movement, identifying morning, midday, and evening peak hours (i.e., 7 am–9 am, 12 pm–1 pm, and 4 pm–6 pm) and non-peak hours (i.e., 10 am–11 am) for pedestrian and bicycle activities.

According to [22], pedestrian motion patterns vary throughout the week. On weekdays, pedestrian traffic tends to be the busiest, with several peaks. On weekends, it gradually increases throughout the day with a peak in the early afternoon, after which it gradually decreases. Moreover, each day has its peculiarities, e.g., evening and night hours are typically more active on Fridays and weekends than on other weekdays [21]. Similar observations are provided in [23,24], where it is shown that during working hours, device activity in the office is high, while on weekends, holidays, evenings, and nights, device activity is rather low. Furthermore, device motion patterns are correlated with population growth [21].

**Device-dependent mobility.** According to [22], bicycle activity has peak hours between 8 am and 10 am, whereas pedestrian activity starts an hour earlier. At approximately 2 pm, there is a second peak during the lunch break that ends at 4 pm, which is different from the pedestrian motion with peaks at 12 pm–1 pm and 4 pm–6 pm. The evening peak hours are consistent with the pedestrian traffic. The weekend bicycle patterns resemble weekdays, peaking at 4 pm and 8 pm [22]. In contrast, vehicle movement, characterized by higher speeds and predetermined routes, differs from pedestrian and bicycle mobility due to road constraints and urban congestion [14].

As a further example, extended reality (XR) usage results in diverse movement patterns compared to mobile phone usage due to unique content presentation and navigation experience [13]. XR causes movements with shorter stride lengths, longer stance times, and higher

speed variability. Moreover, the pace changes are substantially lower in the case of XR during dual-tasking compared to the motion pattern of users with mobile phones, validating XR stability and multitasking sustainability [25].

**Space-dependent mobility.** Mobility patterns also vary based on location and can be highly unpredictable [22]. Despite fluctuations over time and location, there is a geographical pattern in the motion activity and information flows [26].

Bicycle traffic is but one example of a space-dependent motion pattern [22], where local activity cycles differ from global patterns. The activity peaks near a university occur from 8 am to 1 pm, which is typical for institutions that offer morning classes or workplaces. Moreover, room occupancy and mobility patterns inside universities differ depending on room type (meeting room, laboratory, or office) [27]. Afternoon peaks from 3 pm to 4 pm result from lunch breaks or shifts between the morning and afternoon classes. Another spike appears after 8 pm likely linked to the nearby bars and restaurants. A location near a hospital and office buildings shows increased activity around 8 am driven by steady work schedules rather than fluctuating university course start times. Residential districts, however, exhibit the opposite behavior, where people leave in the morning and return later in the afternoon or evening. On weekends, areas near malls display a unique bimodal distribution due to the attraction of afternoon visitors.

**Situation-dependent mobility.** Analyzing situation-dependent mobility, i.e., during emergencies, is essential for SIoT research. In this case, unexpected activity peaks may occur due to specific conditions that trigger a response. In contrast, there may be complete cessation of device movement during lockdown. However, there is a lack of data on spatio-temporal movement patterns during catastrophes and other rare events [26].

**Summary.** Various factors, such as time, device type, space, and circumstances, influence the mobility profile of generic devices and of SIoT devices as well. Demographics, such as the age and family status of human users, may also play a role in shaping mobility patterns [28]. These complex behaviors are expected to significantly impact the placement of the SDT at the network edge and call for adaptive migration algorithms that properly capture such dynamics.

## 2.2. DT placement and migration at the edge

**DT basics.** Initially conceived by the National Aeronautics and Space Administration (NASA) in the '60 to support the early space programs, the concept of DT has recently been revamped in many industries, ranging from manufacturing to automotive, passing by the telco domain [4,29]. A DT is a virtual counterpart of a physical object. Typically, a DT architecture mainly consists of three main components: (i) the physical object, e.g., a vehicle, a smartphone, a sensor; (ii) the corresponding DT in charge of modeling, describing, monitoring, and predicting the behavior of the physical object; (iii) the bidirectional interactions between the physical entity and its digital representation, a.k.a. *intra-twin* communications. Through these interactions, the physical object regularly updates its virtual counterpart about its status and can also be controlled/configured by it [15].

Thanks to a DT providing a living copy of the physical object and cognitive capabilities, its entire lifecycle, from design to operation, can be monitored precisely, predicted, improved, and optimized where and when needed to provide the users with the best possible experience, without affecting the physical domain. For instance, a DT in the manufacturing context can predict when equipment is wearing down or needs repair, improve the machine's performance, extend its lifetime, and learn how to redesign to do even more [29].

DTs must be consistently synchronized with the corresponding physical entities. Moreover, raw and heavy data often need to be exchanged, e.g., to build the DT model. Hence, it is crucial to make intra-twin communication fast and efficient [15].

**Literature overview.** Placing DTs at the edge can facilitate faster and more efficient intra-twin communications. Edge placement has been catalyzing the interest of the research community. For example, a cloudlet placement strategy has been proposed in [16], which accounts for the cost of edge server deployment and the delay between physical objects and their DTs.

In the case of device mobility, a DT placement decision can be stale and unable to meet the intra-twin latency constraint. Hence, DT migration should be triggered. For instance, in Fig. 1, the SDT of user  $a$ ,  $DT(a)$ , is placed at the edge server,  $MEC\ host_1$ , reachable through Access Point (AP)  $AP_1$ . When user  $a$  moves and connects to a new AP,  $AP_2$ , keeping  $DT(a)$  at the same edge server may result in a long intra-twin communication latency. In such a case, SDT migration needs to be triggered to the new closest edge server,  $MEC\ host_2$ . However, the mobility of the devices alone is not sufficient to trigger migration. To enable migration towards the target edge server, its capabilities should be sufficient to host the DT, and the incurred network overhead for transferring the DT should also be accounted for.

Hence, deciding *when* and *where* migrating a DT is a challenging issue. In [17], the DT placement and migration problem is formulated and solved through a Deep Reinforcement Learning (DRL)-based algorithm aimed at reducing average system latency. Moreover, since training a complete DRL model can incur long latency and heavily consume computing resources, a DT migration method based on transfer learning, where decisions are triggered at fixed time intervals, is proposed.

In [18], online DT migration and resource management in multi-tier computing networks are addressed. The goal is to minimize the data synchronization latency between the DT and its physical counterpart by considering time-varying network conditions and user mobility. The framework utilizes convex optimization methods to determine the optimal allocation of communication and computation resources at each edge server, whereas a decentralized partially observable Markov decision process formulation is employed to address the migration problem.

DTs placement and migration are also addressed in [30] to account for both mobility of users and dynamics of resource demands. There, a game-theoretical approach is leveraged, and a Shapley value-based scheme is designed. Dynamic DT placement is considered in [31], where efficient algorithms are devised to maximize user service satisfaction measured through the Age of Information (AoI) of DT data. Similarly, in [32], the focus is on mobility-aware continuous service provisioning in a DT-assisted MEC network that involves mobile objects and is implemented through the placement of DTs. DTs may also need to interact among them whenever they accomplish complex tasks in a cooperative manner [15]. In such cases, placement decisions should also account for the latency incurred in *inter-twin* communications.

A service entity placement problem, which considers activation, placement, proximity, and co-location costs, has been presented in [20]. The DT social aspect has been considered in our previous work in [8], where a mathematical model for the optimal SDT placement at the edge, which minimizes both the delays (i) between physical devices and their virtual counterparts and (ii) among the SDTs of friend devices, has been formulated and an efficient heuristics developed for it. There, the relocation of the SDTs is triggered at fixed time intervals, regardless of the device mobility, which may result in wasted resources for unnecessary migration events. In this work, we aim to overstep this limitation in [8] while keeping, instead, the same efficient heuristic algorithm.

ML techniques can be used to predict the mobility of the physical devices and proactively determine the optimal destination of their corresponding DTs, as theoretically argued in [15,19]. Then, in [33], a service migration scheme in DT-enabled MEC networks is proposed, which leverages traffic prediction and federated deep reinforcement learning to optimize service migration decisions and improve cost efficiency. However, when considering social DTs, migration of an SDT

**Table 1**  
Comparison of DT placement and migration solutions.

Ref., year	Placement	Migration	Orchestration of DTs	Optimization	Heuristics	Motion prediction	Social features	Real dataset
[20], 2018	✓	✗	✗	✓	✓	✗	✓	✓
[16], 2019	✓	✓	✗	✓	✓	✗	✗	✓
[11], 2020	✓	✗	✗	✗	✓	✗	✓	✓
[17], 2021	✓	✓	✗	✓	✓	✗	✗	✗
[8], 2022	✓	✓	✓	✓	✓	✗	✓	✓
[30], 2023	✓	✓	✗	✓	✓	✗	✗	✗
[31], 2024	✓	✗	✗	✓	✓	✗	✗	✗
[19], 2023	✓	✓	✗	✗	✗	✓	✗	✗
[15], 2023	✗	✓	✗	✗	✗	✓	✗	✗
[18], 2024	✓	✓	✗	✓	✓	✗	✗	✓
[32], 2024	✓	✗	✗	✓	✓	✓	✗	✗
<b>Our work</b>	✓	✓	✓	✓	✓	✓	✓	✓

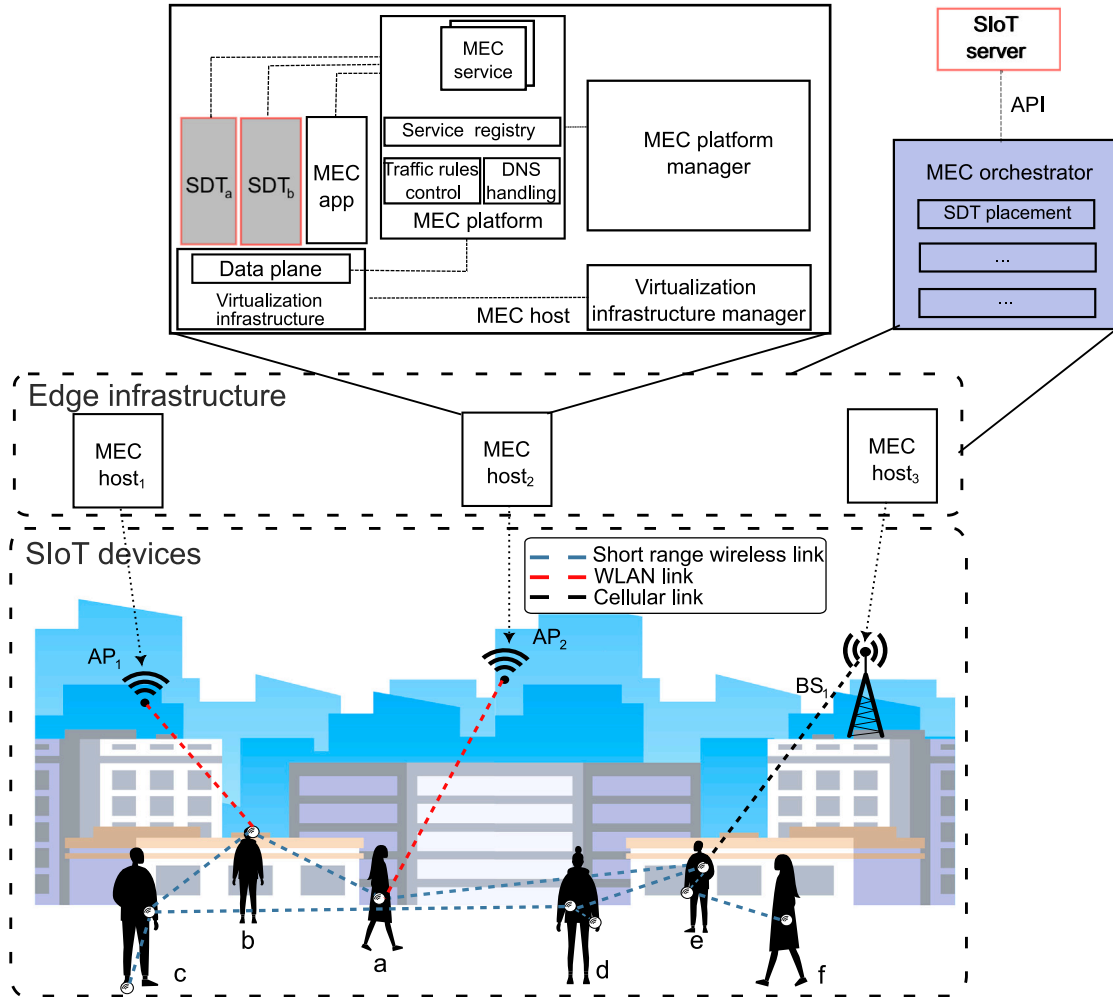


Fig. 1. The SIoT-Edge framework.

may require re-locating the SDTs of its friend devices to meet the targeted navigability latency of the SIoT network. To the best of our knowledge, the potential of ML in making more socially conscious migration decisions has not been well investigated, hence motivating the contributions of this study. Table 1 presents a comparison between the most closely related research and our work.

### 3. Proposed framework

Our work builds upon the SIoT-edge framework, which is graphically sketched in Fig. 1 and initially proposed in [8]. It is aligned with the European Telecommunications Standards Institute (ETSI) Multi-access Edge Computing (MEC) architecture [34]. In this context, the

placement of SDTs is decided by the MEC orchestrator handling a certain number of edge servers. In this work, we extend the previous framework in [8] to more judiciously trigger SDT migration decisions by augmenting SDTs with cognitive capabilities to predict device mobility and support the MEC orchestrator.

In the following, the main pillars of the proposed framework are shortly recalled, and the conceived enhancements are described. The main notations are collected in Table 2.

#### 3.1. SIoT devices and SDTs

SIoT devices can connect in many ways, e.g., through device-to-device (D2D) links to other devices or through the 5G Radio Access

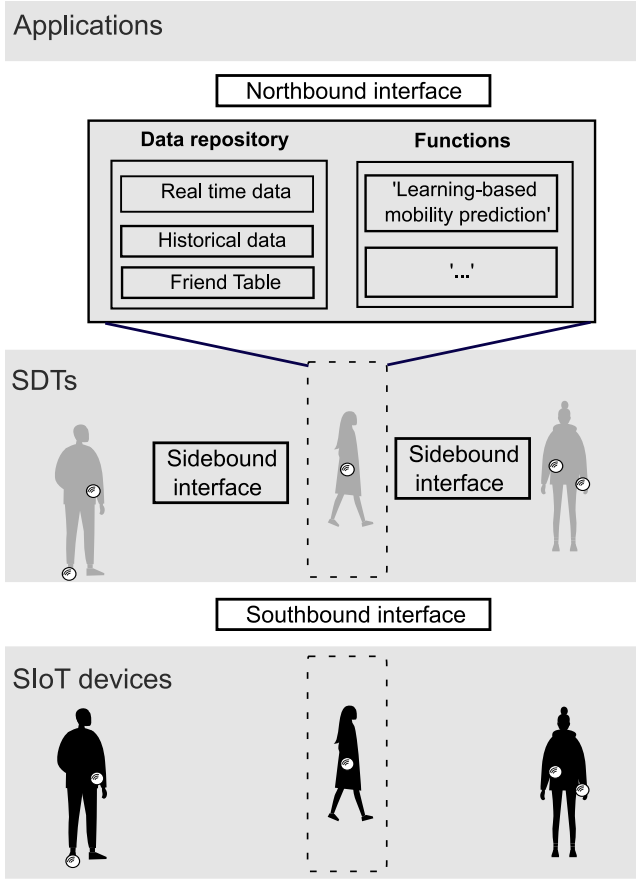


Fig. 2. Main components and interfaces of SDT.

Network (RAN) facilities, such as Base Stations (BSs)/APs, to remote entities. Each SIoT device is paired with its SDT, which describes the physical counterpart and provides it with additional storage and computing capabilities.

The main components of the SDT are reported in Fig. 2. The southbound interface allows the SDT to collect data from the physical counterpart. Such data may encompass sensing, storage, computing, and networking capabilities, as well as the running status of the corresponding resources. Among the collected data, the position of the physical device is periodically transmitted to the SDT. The data repository stores both real-time and historical data retrieved from the physical counterpart. Such data are used to describe the physical object, to model its status, and to predict its future behavior.

Thanks to the data repository, the SDT can cache and aggregate the raw data transmitted by the IoT device before IoT applications can process it. In doing so, the SDTs can interact with remote applications on behalf of the paired physical devices by relieving their pressure.

Compared to a conventional DT, the SDT also stores information on the social links created by the physical device based on the SIoT paradigm, which we refer to as *Friend Table* in the data repository. For each friend device, it contains information on the device type and the SIoT relationship. It could be: *Co-Location Object Relationship* (C-LOR) established among devices located in the same place, *Ownership Object Relationship* (OOR) defined among devices that belong to the same owner, *Parental Object Relationship* (POR) defined among devices belonging to the same production batch, *Social Object Relationships* (SOR) established due to sporadic or continuous contact of users/devices [2].

An IoT device may exploit friendship information because it may query friend devices, discover services offered by them, and/or exchange data with them. In our design, the SDT itself can interact

with its peers on behalf of the physical device (e.g., to implement distributed cooperative learning processes for better predictions). In particular, the sidebound interface is leveraged for such a purpose, and the information stored in the Friend Table is exploited to discover friend SDTs.

Each SDT includes a set of functions that are responsible for carrying out its tasks also according to the envisioned applications and mainly devoted to describing and predicting the behavior of the corresponding physical device. Among them, a specific module is meant to predict the future location of the paired SIoT device.

### 3.2. Edge infrastructure

Each SDT is designed and deployed as a virtualized application, i.e., *MEC app* (e.g., through containers) in the MEC architecture, and instantiated in edge servers. The latter, referred to as MEC hosts, may be associated with BSs/APs.

SDT placement at the edge ensures low-latency interactions with physical pairs [35]. Moreover, by running at the edge, the SDT can more easily interact with other SDTs and also with natively provided context-aware edge services, as those offered by the ETSI MEC architecture [34].

The core of the edge infrastructure is the MEC orchestrator. It is aware of the resources and capabilities of the edge network and determines the most suitable MEC hosts for instantiating virtualized applications based on latency, processing requirements, available resources, and mobility conditions.

### 3.3. Framework formulation

The MEC orchestrator can interface with the SIoT server through an Application Programming Interface (API) to obtain information about social links among devices. It orchestrates the proper placement of SDTs according to the time-efficient enhanced Social-aware Closest Edge Placement (eSoCEP) algorithm that we proposed in [8]. The operation is executed in a discrete-timing manner using a sequence of time slots  $t \in \mathcal{T} = \{0, \dots, T\}$  in order to capture the mobility features and enable dynamic decision-making among the sets of SIoT devices and edge servers.

We aim to jointly minimize the intra-twin communication latency and the latency among SDTs of friend devices while ensuring that delay bounds for intra-twin communications are met, whenever requested, and effective utilization of the available resources for heterogeneous SDT demands is guaranteed. Note that minimizing the first latency contribution results in placing SDTs as close as possible to their physical counterparts, while also reducing the amount of traffic that passes through the edge network segment. The second latency contribution accounts for the fact that SDTs may often interact with each other to provide (low-latency) SIoT-based services.

This can be mathematically formulated as follows [8]:

$$\begin{aligned} \min \quad & \sum_{i \in V_p} \sum_{k \in V_s} x_{ik}(t) L_{ik}(t) \\ & + \sum_{i \in V_p} \sum_{k \in V_s} \sum_{j \in V_p} \sum_{l \in V_s} x_{ik}(t) x_{jl}(t) p_{ij}(t) L_{kl}(t), \end{aligned} \quad (1)$$

s.t.

$$x_{ik}(t) \in \{0, 1\}, \quad \forall i \in V_p, \quad \forall k \in V_s, \quad \forall t \in \mathcal{T}, \quad (2)$$

$$\sum_{k \in V_s} x_{ik}(t) = 1, \quad \forall i \in V_p, \quad \forall t \in \mathcal{T}, \quad (3)$$

$$L_{ik}(t) \leq L_{\max_i}, \quad \forall i \in V_p, \quad \forall k \in V_s, \quad \forall t \in \mathcal{T}, \quad (4)$$

$$\sum_{i \in V_p} \frac{x_{ik}(t) CPU_i(t)}{a CPU_k} \leq THR_{CPU}, \quad (5)$$

$$\sum_{i \in V_p} \frac{x_{ik}(t) D_i(t)}{a D_k} \leq THR_D, \quad (6)$$

**Table 2**  
Main notations.

Framework parameters	
$\mathcal{T}$	Time horizon
$V_P$	Set of SIoT devices
$V_S$	Set of SIoT edge servers
$x_{ik}$	Binary variable representing whether SDT of device $i$ is placed into edge server $k$
$x_{jl}$	Binary variable representing whether SDT of device $j$ is placed into edge server $l$
$L_{ik}$	Latency between device $i$ and its SDT
$L_{kl}$	Latency between each pair of edge servers $k$ and $l$
$p_{ij}$	Probability that characterizes the intensity of data exchange between SIoT devices $i$ and $j$
$L_{\max_i}$	Maximum tolerated latency between physical device $i$ and its SDT
$aCPU_k$	CPU capability of edge server $k$
$aD_k$	Disk capability of edge server $k$
$aRAM_k$	Memory capability of edge server $k$
$CPU_i(t)$	CPU requirement to execute the SDT of device $i$
$D_i(t)$	Disk requirement to execute the SDT of device $i$
$RAM_i(t)$	Memory requirement to execute the SDT of device $i$
$THR_{CPU}$	CPU utilization threshold
$THR_D$	Disk storage utilization threshold
$THR_{RAM}$	RAM utilization threshold
Learning-specific parameters	
$S$	Dataset size
$n_{\text{samples}}$	Number of training samples
$n_{\text{features}}$	Number of features
$n_{\text{estimators}}$	Number of trees in ensemble algorithms
$d_{\text{depth}}$	Depth of the tree
$r_{\text{success}}$	Number of successive models
$n_{\text{vectors}}$	Number of support vectors
Sliding window	Size of sliding widow
$n_{\text{hidden}}$	Number of hidden units of LSTM layer

$$\sum_{i \in V_P} \frac{x_{ik}(t)RAM_i(t)}{aRAM_k} \leq THR_{RAM}, \quad (7)$$

where  $x_{ik}(t)/x_{jl}(t)$  is a placement decision variable that takes the value 1 if SDT of device  $i/j$  is mapped to edge server  $k/l$ , otherwise  $x_{ik}(t)/x_{jl}(t) = 0$ , and  $L_{ik}(t)$  represents the latency between device  $i$  and its SDT, while the latency between each pair of edge servers  $k$  and  $l$  is denoted by  $L_{kl}(t)$ . The probability  $p_{ij}(t)$ , where  $0 \leq p_{ij}(t) \leq 1$ , characterizes the intensity of data exchange between SIoT devices  $i$  and  $j$ , and is linked to the connections in the SIoT.  $L_{\max_i}$  denotes the maximum latency between physical device  $i$  and its SDT deployed at edge server  $k$ . The CPU, disk, and memory capabilities are denoted by  $aCPU_k$ ,  $aD_k$ ,  $aRAM_k$ , respectively. The requirements to execute SDT of device  $i$  correspond to  $CPU_i(t)$ ,  $D_i(t)$ , and  $RAM_i(t)$ , whereas the threshold values for CPU, disk storage, and RAM utilization are denoted by  $THR_{CPU}$ ,  $THR_D$ , and  $THR_{RAM}$ , respectively.

Constraint (3) ensures SDT placement without replication, while Constraint (4) represents physical device-SDT proximity constraints. Finally, Constraints (5)–(7) ensure efficient resource utilization and prevent server overload.

This problem is widely acknowledged to have NP-hard complexity, which renders its solution computationally intensive through exhaustive search. Therefore, we employ our graph-based heuristics (eSoCEP) proposed in [8], which tackles the complex problem of optimally placing the SDTs of devices on edge servers to minimize communication costs, as per (1). It breaks down the problem into smaller and more manageable pieces. Specifically, it identifies tightly connected groups of SIoT devices that communicate frequently with each other. Within each group, it identifies the most efficient way to connect SDTs to servers by using a spanning tree. The algorithm then repeats this process for each group of devices, prioritizing the most critical connections within each group. Finally, it assigns servers to each SDT based on the results from the previous steps.

The algorithm has to process  $n$  devices in groups and within each group, as well as to analyze the communication patterns between all devices (up to  $n$  devices). Here,  $O(n)$  is the complexity due to the cycle over all vertices of the graph, which is, in the worst case, equal to the number of SIoT devices ( $n$ ). Moreover, the algorithm searches for the

largest connected component in terms of the number of communications between all vertices (SIoT devices). The upper bound for it is  $O(n)$ . As a result, this nested processing over potentially many groups leads to  $O(n^2)$  complexity [8].

Complexity also affects the processing time at the edge server. It is worth noticing that experiments reported in our previous work [8] already demonstrated that the processing time required at the edge server to run eSoCEP for a large number of devices is up to 93% lower than the time needed to find the optimal solution through a standard solver, showing that the proposed heuristics also scales well with the number of devices, while providing near-optimal performance.

#### 3.4. Learning-triggered SDT migration

In [8], the heuristics deciding the placement of SDTs and, hence, their potential migration, was run in the MEC orchestrator at fixed time intervals regardless of device mobility. However, this choice may not be efficient in a dynamic environment. The small cell radius of wireless networks transmitting at the millimeter-wave (or upcoming terahertz) bands may cause frequent handovers and possible consequent switching of SDTs among MEC hosts to ensure low-latency intra-twin communications. However, changing edge servers at fixed intervals or simply doing this upon cell switching is inefficient from a network perspective and can waste resources without necessarily improving the service delivery.

In this work, in contrast to [8], heuristics execution at the MEC orchestrator is triggered by the SDTs themselves only when the predicted mobility patterns for the paired SIoT devices may adversely affect the intra-twin communication latency. Specifically, each SDT is equipped with ML/DL capabilities that enable the prediction of mobility of the corresponding physical device based on previously recorded mobility patterns.

Leveraging its embedded ML/DL model (Fig. 2), each SDT periodically infers the next predicted location of the paired physical device, which might imply a change of network attachment. In such a case, it checks whether its actual edge placement will still satisfy the targeted latency constraints for intra-twin communications. Only if this is not the case, then the SDT triggers the MEC orchestrator to run the heuristics responsible for the SDT relocation according to

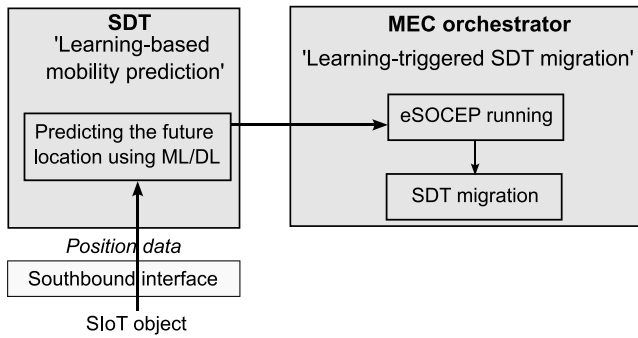


Fig. 3. Visual overview of the proposal.

the new predicted location. It should be noted that the relocation may consequently involve several SDTs. This is because, besides intra-twin communication latency, inter-twin communication latency can also be affected by the relocation of the SDTs of devices tied by social relationships.

All SDTs periodically report their predicted positions to the MEC orchestrator by piggybacking the SDT relocation trigger only when needed. The reporting frequency may be set according to the specific requirements of the services and dynamics of the ML/DL algorithms embedded within SDTs.

### 3.5. Learning-based mobility prediction

The SDT includes a specific module for predicting the future location of its paired SiIoT device. To do so, SDT relies on ML/DL techniques, such as LSTM, RF, SVR, XGBoost, and  $k$ NN, which are suitable for time series forecasting [36–39]. This class of models well matches the problem at hand, thanks to their ability to model sequences of observations over time and capture complex patterns in the motion data [40,41]. The ML/DL model is trained by using both historical data about the mobility pattern in a given area and also real-time data collected through the SiIoT device.

*Significance of mobility prediction:* Mobility prediction can be applied for a variety of real-time applications, such as selecting the best access point [42], allocating resources effectively [43], optimizing load balancing [44], and controlling seamless handovers for users [45]. In the context of SDTs, as we demonstrate in Section 4.3, the mobility prediction of virtual devices facilitates SDTs' placement and migration to reduce the service discovery latency.

Cooperative sensing and tracking technologies can be leveraged to improve the accuracy of the position information. For example, high-level sensing and localization can be obtained from low-level raw measurements, such as channel state information (CSI) and received signal strength indicators (RSSI) between peers through D2D communications and local area networks, i.e., Wi-Fi [46,47]. D2D links can also facilitate localization and sensing through cooperative positioning, where peers can exchange necessary data, such as common physical layer estimates and position information, to increase the positioning accuracy [48].

Fig. 3 depicts a schematic representation of the proposal, which includes two main components: learning-triggered SDT migration and learning-based mobility prediction, discussed in Sections 3.4 and 3.5, respectively.

## 4. Performance evaluation

We conduct a two-step evaluation study to assess the performance of the proposed learning-powered SDT migration strategy. We first compare different ML/DL techniques used in step 1 for the prediction of SiIoT device mobility to identify the best-performing technique. As

a software environment, Python is leveraged for this stage due to its extensive libraries for ML/DL. The eSoCEP heuristics, run in step 2, is implemented in a MATLAB environment, which offers strong capabilities for graph-based algorithms. The execution of the algorithm is triggered by SDTs running device mobility predictions once the intra-twin communication latency is at risk.

### 4.1. Scenarios and settings

We consider the reference scenarios as detailed in the following.

*Scenario 1-Mixed.* The simulation scenario corresponds to a 4 km<sup>2</sup>-large area of the city center of Santander in which 113 heterogeneous devices, including smartphones, cars, tablets, smart fitness devices, and smartwatches, move and establish social relationships, as per the dataset in [12]. This dataset tracks interactions among IoT devices in real-world scenarios<sup>1</sup> and employs the Small World in Motion (SWIM) mobility model [49].

Each device in the dataset establishes social relationships of types OOR, C-LOR, SOR, and POR distributed with percentages of 50%, 15%, 21%, and 14%, respectively. We presume the probability values  $p_{ij}(t) = 1$  for OOR connections and  $p_{ij}(t) = 0.1$  for C-LOR, SOR, and POR [8].

We assume a 3GPP-compliant hexagonal grid with 8 BSs, each co-located with an edge server. Each device has an SDT implemented as a container at the edge. SDTs are associated with four types of containers based on CPU and RAM demands. High-CPU medium (2000 MIPS/0.85 GB) and extra-large (2500 MIPS/3.75 GB) SDTs are for mobile phones and cars with autonomous navigation, respectively, while smartwatches, sensors, tablets, smart fitness gadgets, and printers require small (1000 MIPS/1.7 GB) or micro (500 MIPS/613 MB) instances.

*Scenario 2-Pedestrians.* The simulation scenario extends over 250 m × 500 m, featuring 8 BSs with an intersite distance of 120 m [50]. To emulate real-life user behavior in diverse application scenarios, we implement a pedestrian flow simulation based on the social force-based model of human behavior [25,51]. Pedestrian motion is simulated at a speed of 3 km/h.

The interactions among the 113 SiIoT devices are randomly generated, maintaining the same probability values as in *Scenario 1-Mixed*. The demands of the SDTs and delay constraints between a physical device and its SDT are also consistent with those in *Scenario 1-Mixed*. The difference lies in SDTs being associated with a single container type, i.e., extra-large (2500 MIPS/3.75 GB) SDTs since pedestrians are assumed to carry mobile phones.

As regards the settings common to both scenarios, edge server CPU, disk, and RAM capabilities correspond to 24000 MIPS, 2 TB, and 24 GB, respectively. The disk demands of SDTs are uniformly distributed in [10,50] GB, whereas the delay between a physical device and its SDT must be within 1 – 10 ms.

The latency between devices and edge servers, as well as the latency between edge servers, is directly proportional to the physical distances between them [16,20]. This relationship is established using a coefficient for the distance-to-latency mapping of 3.33 ms/km [52].

The dataset contains  $n_{\text{features}} = 113$  columns representing each device and  $S$  rows representing the number of zones where the device is located at each timestamp. Since device mobility depends on the time of the day and rush hours, among other factors discussed in Section 2.1, we take the dataset size of  $S = 1000$  for scenario 1 and  $S = \{1000, 2000\}$  for scenario 2, assuming that after 1000 timestamps the model will be retrained.

The simulations for both scenarios cover a 5-hour time interval (corresponds to  $S = 1000$ ), within which, according to the proposed framework, the placement and migration of SDTs are triggered on-demand by running the eSoCEP heuristics when utilizing the ML/DL algorithm at SDTs to predict mobility. Position data are sampled every

<sup>1</sup> <https://www.fiware.org>.

**Table 3**  
Theoretical complexity analysis of ML/DL models.

Algorithm	Type	Training complexity	Inference complexity
LSTM layer	DL	$O(4 \text{ sliding window} \cdot n_{\text{features}} \cdot n_{\text{hidden}} + 4n_{\text{hidden}}^2 + 3n_{\text{hidden}} + n_{\text{hidden}} \cdot n_{\text{features}})$	$O(\text{sliding window} \cdot n_{\text{features}} \cdot n_{\text{hidden}})$
RF	ML	$O(n_{\text{estimators}} \cdot n_{\text{samples}} \cdot \log(n_{\text{samples}}) \cdot d_{\text{depth}})$	$O(n_{\text{estimators}} \cdot d_{\text{depth}})$
SVR	ML	$O(n_{\text{samples}}^2 \cdot n_{\text{features}} + n_{\text{samples}}^3)$	$O(n_{\text{vectors}} \cdot n_{\text{features}})$
kNN	ML	$O(1)$	$O(n_{\text{samples}} \cdot n_{\text{features}})$
XGBoost	ML	$O(n_{\text{estimators}} \cdot n_{\text{samples}} \cdot \log(n_{\text{samples}}) \cdot d_{\text{depth}} \cdot r_{\text{success}})$	$O(r_{\text{success}} \cdot d_{\text{depth}})$

1s. We analyze the settings with  $n_{\text{features}} = 113$  devices and  $S$  samples that reflect their current locations. We perform training and validation tests on a standard laptop PC equipped with 16 GB of RAM and an Intel Core i7-1260P CPU @ 2.10 GHz and run the code using Python 3.10.6.

#### 4.2. Mobility prediction using ML/DL techniques

We compare the performance of the following ML/DL techniques to identify the one to be implemented at SDTs:

- LSTM is a type of Recurrent Neural Network (RNN) that models time-series data by learning temporal dependencies and patterns. Due to its ability to capture long-term dependencies and handle variable-length input sequences, LSTM is widely used for mobility prediction [40]. Our model consists of the LSTM layer with  $n_{\text{hidden}} = 16$  hidden units and Rectified Linear Unit (ReLU) activation function, the sliding window of length 50, one fully connected hidden layer with 256 units and ReLU activation function, and one fully connected layer with  $n_{\text{features}} = 113$  units. The model is compiled using Mean Absolute Error (MAE) as the loss function, Adam optimizer, and 40 epochs with a unit batch size.
- Random Forest is an ensemble learning technique that uses decision trees to build multiple models and aggregate their predictions. Random Forest handles noisy and high-dimensional data, which is suitable for motion prediction and time-series forecasting [39]. We implement random forest with  $n_{\text{estimators}} = 200$  trees and depth  $d_{\text{depth}} = 4$ .
- SVR is a type of Support Vector Machine (SVM) that can perform regression tasks by identifying the optimal hyperplane that best fits the data by handling non-linear relationships between input and output variables [53]. We implement SVR with the radial basis function kernel, regularization parameter of 1, and epsilon 0.1.
- kNN can be applied to motion prediction by utilizing historical motion data (e.g., previous positions, velocities) as the training set, where similar past motions help predict future movements based on proximity to neighboring motion patterns, making it suitable for trajectory forecasting or behavior prediction in scenarios like object tracking, vehicle motion, or human activity recognition [54]. We implement kNN, utilizing 5 nearest neighbors.
- XGBoost is an ensemble learning method that uses gradient boosting to build a predictive model by combining the outputs of multiple individual models, typically decision trees, in an additive manner. It works by sequentially improving upon the mistakes of previous models to create a more accurate and robust final model [41]. We implement XGBoost with  $n_{\text{estimators}} = 100$  estimators and depth  $d_{\text{depth}} = 6$ .

The complexity of the considered ML/DL algorithms is summarized in Table 3. It is important to note that the nested nature of the operations in the LSTM model and the dependence on the input sequence length (i.e., sliding window) makes it difficult to express the exact complexity in a single big- $O$  notation. However, the training and inference complexities are primarily determined by the terms listed in Table 3.

We compare the performance of the above ML/DL models under different *train/test* split ratios, meaning that *train%* of the mobility dataset

**Table 4**  
Training and inference times, 90/10% train/test split ratio, for both considered scenarios.

Dataset size	Mixed		Pedestrians		Mixed		Pedestrians	
	1000	2000	1000	2000	1000	1000	2000	
		Training time			Inference time			
		[s]	[s]	[s]	[s]	[s]	[s]	
LSTM	183.595	205.771	429.740	0.056	0.052	0.051		
RF	0.572	0.592	1.388	0.008	0.007	0.008		
SVR	0.109	0.229	0.823	0.012	0.009	0.013		
kNN	0.037	0.034	0.039	0.363	0.412	0.256		
XGBoost	2.124	2.006	3.384	<b>0.001</b>	<b>0.001</b>	<b>0.001</b>		

is used for training and *test%* for testing. A good quality prediction can be obtained by updating the model for every 1000 samples (as demonstrated below). We employ an initial random seed of 51 to ensure reproducibility.

We measured not only standard metrics based on absolute values, such as MAE, root mean squared error (RMSE), and accuracy measured as 100–mean absolute percentage error (MAPE), but also the coefficient of determination,  $R^2$  score, which measures the ability of the model to fit the data and accurately predict the dependent variable based on the independent ones [55].

As per *Scenario 1-Mixed*, as shown in Table 4, kNN has the shortest training time since there is no traditional training phase when the algorithm explicitly learns a model based on the training data. Instead, the entire dataset itself effectively becomes the model. SVR's training time is 109 ms providing a fast performance, followed by RF (571 ms) and XGBoost (2.12 s), and then LSTM, which has a longer training time (183.6 s) but a more data-adaptable structure. We also provide one-sample inference time to understand how quickly the models can respond to the new mobility data and let each SDT make predictions to trigger a potential migration promptly. In contrast to the training time, the XGBoost model offers the fastest inference speed with latency in the order of 0.7 ms.

Regarding prediction accuracy on unknown data, XGBoost outperforms the other models in terms of RMSE, MAE, accuracy, and  $R^2$  score across a majority of the training dataset sizes, as depicted in Fig. 4. Notably, a training ratio of 0.90 stands out as the optimal choice for achieving the highest accuracy-related scores within the current dataset. In this case, the accuracy of XGBoost exceeds 97%, the highest value among the evaluated techniques, meaning that, on average, the predictions made by the regression model deviate by merely 3% from the actual target values.

Moreover, the  $R^2$  score of the test data approaches 1, indicating that the model can account for  $\approx 95\%$  of the variation in actual target values. This confirms that the model makes accurate predictions, with the remaining 5% of variance due to factors not captured by the model. Comparatively, among the other models, kNN emerges as the second-best performer in terms of accuracy, MAE, RMSE, and  $R^2$  score up to 80/20% of train/test split ratio (and on the whole train/test split ratio interval for RMSE), albeit with notably longer inference times in the order of 0.36 s. SVR exhibits good enough accuracy solely for a training ratio of 0.90, while RF and LSTM lag notably behind XGBoost in terms of accuracy. The training progress of LSTM (that learns iteratively through backpropagation) is presented in Fig. 5. In stradeoffXGBoost

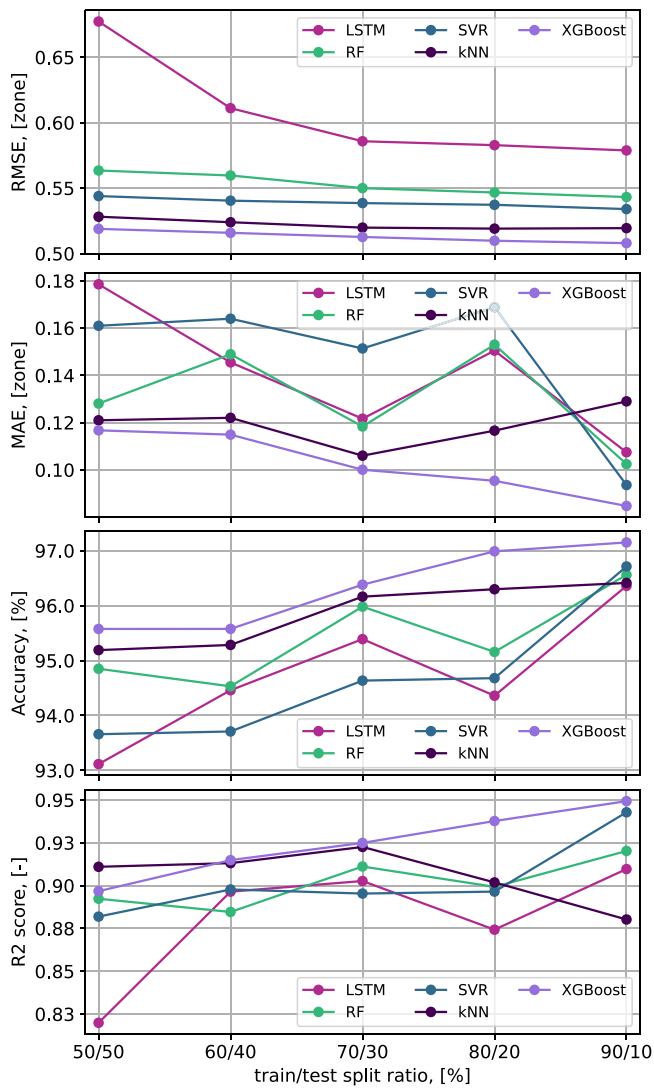


Fig. 4. Scenario 1-Mixed: RMSE, MAE, accuracy, and  $R^2$  score as function of the train/test split ratio, dataset size  $S = 1000$  samples.

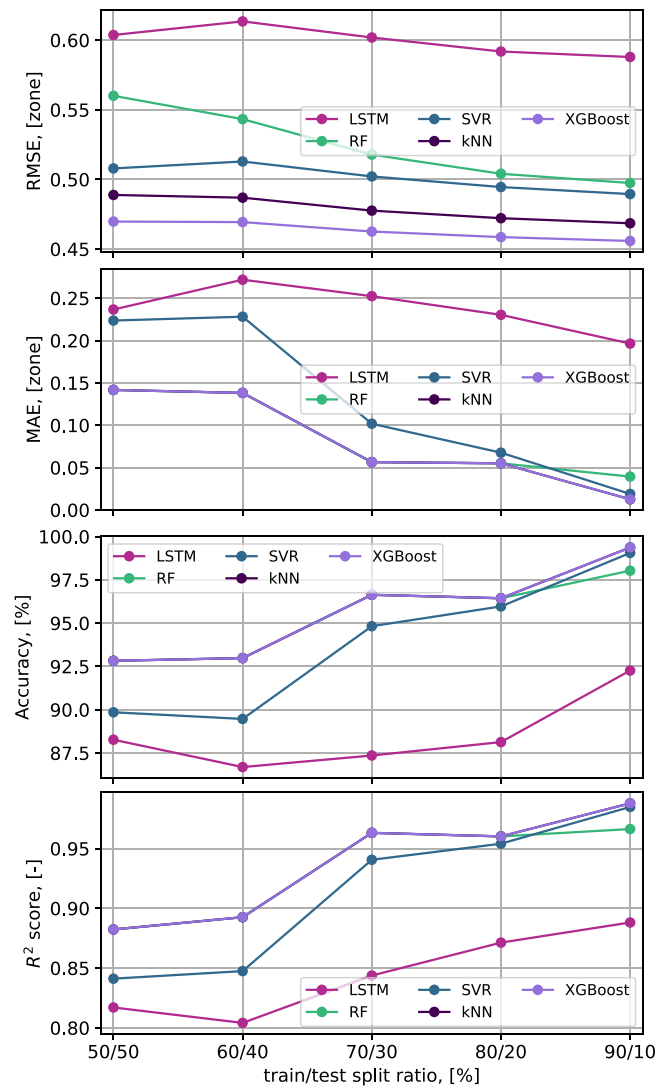


Fig. 6. Scenario 2-Pedestrians: RMSE, MAE, accuracy, and  $R^2$  score as function of the train/test split ratio, dataset size  $S = 1000$  samples.

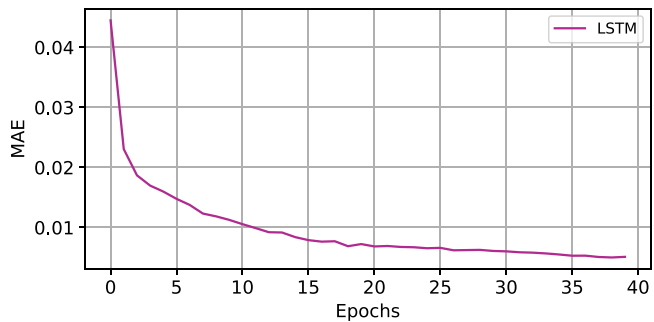


Fig. 5. Scenario 1-Mixed: LSTM loss function value (MAE) as training processes in 40 epochs, 90/10% train/test split ratio, dataset size  $S = 1000$  samples.

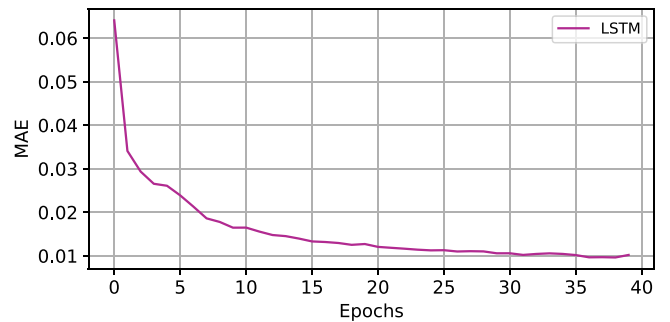


Fig. 7. Scenario 2-Pedestrians: LSTM loss function value (MAE) as training processes in 40 epochs, 90/10% train/test split ratio, dataset size  $S = 1000$  samples.

presents an advantageous balance among the key factors. It delivers a compelling tradeoff between the training time (higher than  $kNN$ , SVR, and RF, yet still relatively small), inference time (the shortest), and accuracy (the highest), making it a notably favorable choice.

In *Scenario 2-Pedestrians*, as presented in Table 4 and Figs. 6 and 7, an intriguing shift unfolds from the previously observed outcomes. Remarkably,  $kNN$ 's performance in terms of MAE, RMSE, accuracy, and

$R^2$  score closely mirrors that of XGBoost, presenting a compelling match in results (up to 6 decimal places). However, despite this near parity,  $kNN$  maintains a drawback in its inference time, which is a crucial factor for time-sensitive applications. Additionally, RF demonstrates results akin to those of XGBoost in *Scenario 2-Pedestrians*, showcasing proximity in MAE, accuracy, and  $R^2$  score. Conversely, SVR exhibits good results, particularly when the training ratio exceeds 0.80, whereas

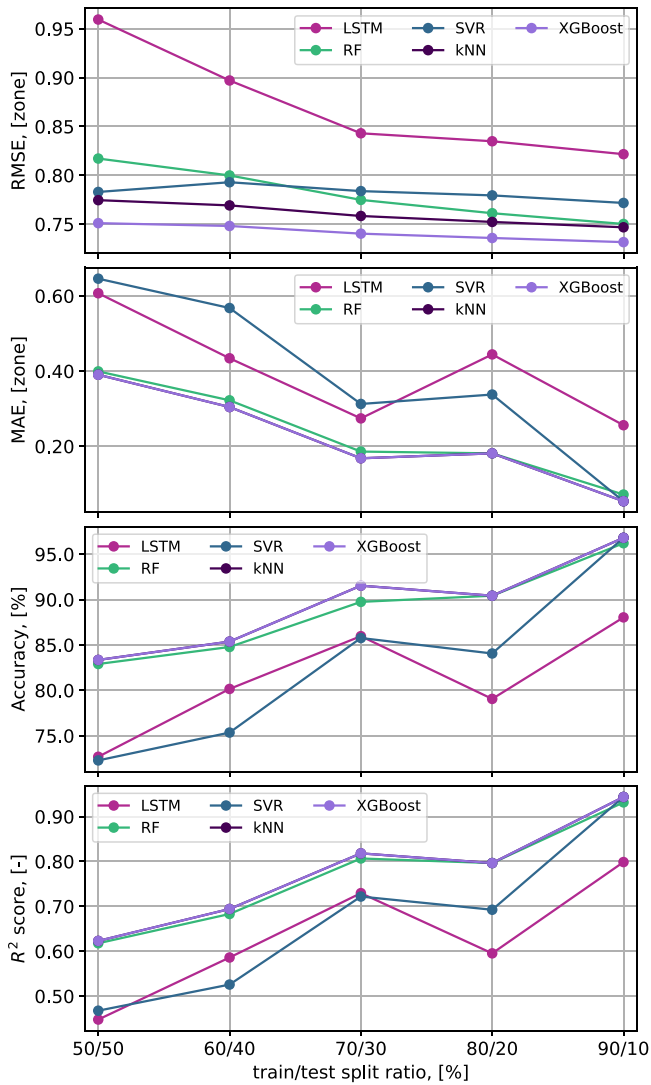


Fig. 8. Scenario 2-Pedestrians: RMSE, MAE, accuracy, and  $R^2$  score as function of the train/test split ratio, dataset size  $S = 2000$  samples.

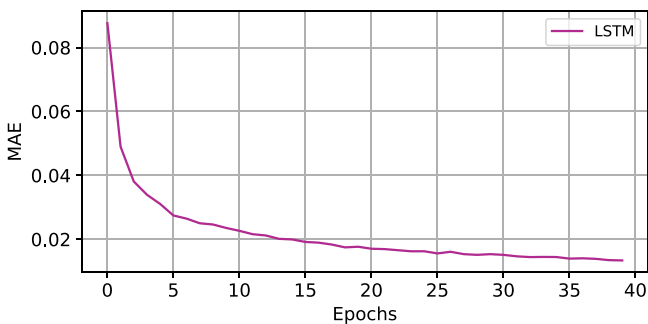


Fig. 9. Scenario 2-Pedestrians: LSTM loss function value (MAE) as training processes in 40 epochs, 90/10% train/test split ratio, dataset size  $S = 2000$  samples.

LSTM continues to reveal limitations, proving that it is less suitable for this specific task.

In addition, given the relatively modest movement speed in pedestrian mobility scenarios, we conduct additional tests with an increased number of samples in the *Scenario 2-Pedestrians* dataset. As previously underscored, our recommendation still advocates retraining the models periodically, specifically at every 1000 sample interval, owing to the

Table 5

5-fold cross-validation. The mean value is provided.

Dataset size	Mixed		Pedestrians		Mixed		Pedestrians		
	1000	1000	2000	1000	1000	2000	1000	2000	
				MAE		RMSE			
LSTM	0.066	0.069	0.142	1.172	0.089	0.184			
RF	0.058	0.043	0.094	3.363	0.056	0.117			
SVR	0.077	0.105	0.216	2.277	0.116	0.234			
kNN	0.052	0.035	0.076	5.369	0.053	0.110			
XGBoost	0.042	0.034	0.077	1.581	0.053	0.111			

dynamic nature of the data. This imperative emerges evident upon examining Figs. 8 and 9 with dataset size  $S = 2000$  samples, wherein a discernible decline in accuracy and  $R^2$  score is apparent alongside an escalation in MAE and RMSE metrics. These observed fluctuations signify the evolving nature of the dataset, underscoring the necessity for regular model updates to sustain optimal predictive performance. Similar to the findings observed in *Scenario 1-Mixed*, it is noteworthy that a training ratio of 0.90 emerges as the optimal choice within the context of *Scenario 2-Pedestrians*. What also remains consistent across both considered scenarios is the undeniable supremacy of XGBoost over other ML/DL techniques.

*Cross validation:* To test how ML/DL models work on unseen data, we provide results for 5-fold cross-validation while considering MAE and RMSE (see Table 5). This involves training models using various subsets of data (5 subsets), subsequently yielding an average output for the final result. This approach also helps mitigate the potential for overfitting, thereby enhancing the overall effectiveness of the model. Results demonstrate that, on average, the models generalize well on different parts of the dataset.

Therefore, based on the results for Scenario 1-Mixed and Scenario 2-Pedestrians for both train/test split and cross-validation, in the following, we leverage the XGBoost algorithm (showing a good accuracy and inference speed) in the SDT to predict the movement of the paired physical device.

#### 4.3. ML-powered SDT migration

We compare the proposal against an approach in [8] representative of *blind* solutions, where SDTs relocation is triggered at fixed time intervals, whose duration  $t$  is equal to 1, 5, 10, 15, 20, 25, and 30 min, irrespective of changes of the network attachment, and leveraging device location retrieved at the previous time interval to take the decision. Our proposal, instead, is analyzed for the mobility prediction interval, which is equal to 1 s.

The following metrics are measured and reported in Tables 6 and 7:

- Number of times the relocation algorithm is run to assess the complexity incurred by the proposal at the orchestrator to trigger the migration of SDTs;
- Number of migrated SDTs calculated as the total number of SDTs migrated from one edge server to another throughout the whole simulation time.
- Network overhead computed as the total amount of data transferred (in GB) from one edge server to another to migrate the SDT container.
- Service discovery latency derived as the sum of the intra-twin communication latency and the latency experienced to reach the SDTs of friend devices one by one.

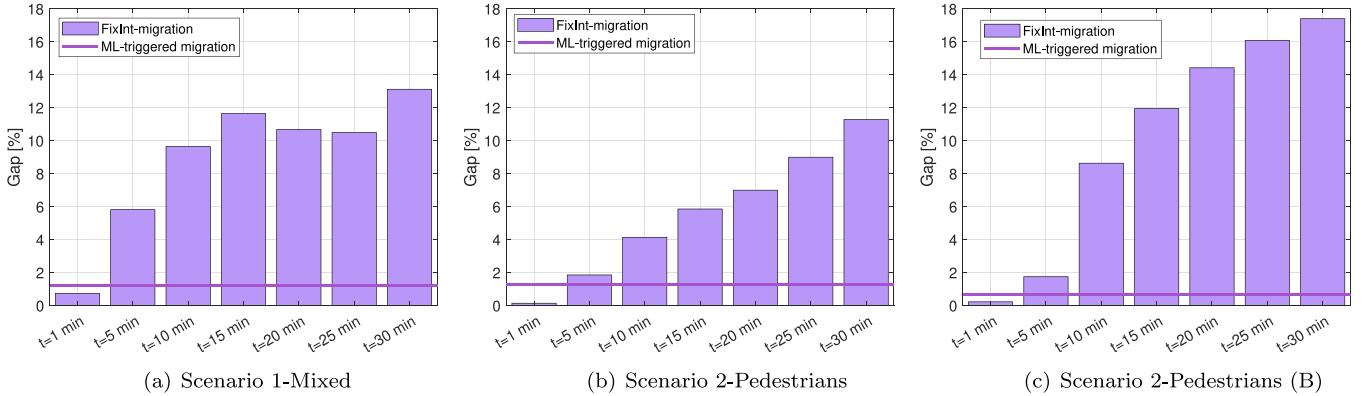
As per *Scenario 1-Mixed*, the shortest service discovery latency is achieved for  $t = 1$  min. Correspondingly, as a side effect, the highest number of migrated SDTs and incurred overhead are measured. Hence, this setting wastes too much network resources. On the other hand, longer time intervals between relocation decisions lower network overhead but at the expense of a higher service discovery latency. Our

**Table 6**  
Migration-related metrics and service discovery latency (Scenario 1-Mixed).

	Runs of relocation algorithm [-]	Number of migrated SDTs [-]	Network overhead [GB]	Service discovery latency [ms]
FixInt-migration ( $t = 1$ min)	300	930	27905	7.65
FixInt-migration ( $t = 5$ min)	60	324	9722	8.03
FixInt-migration ( $t = 10$ min)	30	243	7291	8.32
FixInt-migration ( $t = 15$ min)	20	196	5881	8.47
FixInt-migration ( $t = 20$ min)	15	156	4681	8.40
FixInt-migration ( $t = 25$ min)	12	162	4861	8.39
FixInt-migration ( $t = 30$ min)	10	118	3541	8.59
<b>ML-triggered migration (<math>\bar{t} = 4.9</math> min)</b>	<b>63</b>	<b>363</b>	<b>10 892</b>	<b>7.68</b>

**Table 7**  
Migration-related metrics and service discovery latency (Scenario 2-Pedestrians).

	Runs of relocation algorithm [-]	Number of migrated SDTs [-]	Network overhead [GB]	Service discovery latency [ms]
FixInt-migration ( $t = 1$ min)	300	711	21334	7.01
FixInt-migration ( $t = 5$ min)	60	376	11282	7.13
FixInt-migration ( $t = 10$ min)	30	309	9272	7.29
FixInt-migration ( $t = 15$ min)	20	280	8402	7.41
FixInt-migration ( $t = 20$ min)	15	213	6391	7.49
FixInt-migration ( $t = 25$ min)	12	189	5671	7.63
FixInt-migration ( $t = 30$ min)	10	144	4321	7.79
<b>ML-triggered migration (<math>\bar{t} = 7.2</math> min)</b>	<b>42</b>	<b>342</b>	<b>10 262</b>	<b>7.09</b>



**Fig. 10.** Gap of proposed and benchmark solutions from the ground truth in terms of service discovery latency.

proposal has approximately triggered the relocation of SDTs at every  $\bar{t}$  equal to 4.9 min.

*Scenario 2-Pedestrians* demonstrates a similar behavior to *Scenario 1-Mixed*, with our proposal securing the second position in terms of service discovery latency (following the fixed rerun every 1 min). However, there is a more significant reduction in the number of SDT relocations w.r.t. the fixed rerun every 1 min. Reallocation is triggered approximately every 7 min. Such a trend can be ascribed to the fact that slowly moving devices, those carried by pedestrians, are only considered in *Scenario 2-Pedestrians*. Moreover, the amount of migrated data is roughly half of what is migrated every 1 min.

Such results imply a significantly lower network overhead and, additionally, a lower computing footprint at the orchestrator for executing the SDT migration algorithm.

To further assess the viability of the proposal, we measure the impact of the effectiveness of the prediction on the service discovery latency. To this aim, Fig. 10(a) and (b) report the gaps achieved by the previously compared schemes from the ground truth, where the device movement sampled at every 1 s is *a priori* known, and the SDT migration is always triggered accordingly.

As per *Scenario 1-Mixed*, our proposal exhibits proximity of 98.8% to the ground truth (thus, a gap of less than 2%). The gap of the benchmark from the ground truth, on the other hand, is almost always higher than in our proposal as it ranges from around 6% ( $t = 5$  min) to 13% ( $t = 30$  min). Also, in *Scenario 2-Pedestrians*, our proposal exhibits

a proximity of 98.7% to the ground truth. In contrast, the gap of the benchmark from the ground truth is constantly increasing, comprising approximately 0.1%, 2%, 4%, 6%, 7%, 9%, and 11%, respectively, when the algorithm is rerun every 1, 5, 10, 15, 20, 25, and 30 min. The only exception in both cases is for  $t = 1$  min, but in this case, we have seen from the previous measurements that the migrations performed and the network load for the migrations have approximately halved compared to our proposal.

As a further experiment, we report in Fig. 10(c) the gap achieved in *Scenario 2-Pedestrians* when incorporating different probability values for data exchange between IIoT devices,  $p_{ij}(t)$ , to resemble different social behaviors (denoted as *Scenario 2-Pedestrians (B)*). Specifically, the probability to exchange data is set to  $p_{ij}(t) = 1$  for OOR and SOR connections, whereas it is set to  $p_{ij}(t) = 0.3$  for C-LOR, and to  $p_{ij}(t) = 0.1$  for POR connections. The proposal, in this case, is 99% close to the ground truth.

#### 4.4. Guidelines for practical deployment

In the following, we provide some suggestions on how to deploy the proposed framework in practical settings.

**Social relationships creation and maintenance.** We assume that IIoT devices establish social relationships according to the framework presented in [56]. There, social ties are established and managed with the support of the network service provider (NSP), which could either

be a mobile network operator or a fixed telco provider. In [56], the maintenance and collection of information about the social network to be shared with entities/applications consuming it are handled in a *distributed* manner to reduce the computational burden, grant a good level of privacy, and provide high scalability as the number of SIoT devices increases. In particular, there, information about social relationships are locally kept by each SDT, in the Friend Table, as in our work.

**SDT migration decision.** In our case, the MEC orchestrator receives, via proper APIs, information about social relationships from the NSP and leverages them to make the SDT placement and replacement decisions over the edge servers under its control. The MEC orchestrator then implements the conceived eSoCEP heuristics. The latter one can be implemented in a very short time and it has been shown in [8] to scale very well with the number of SIoT devices.

**SDT implementation and management.** SDTs can be deployed through the Docker container technology as proposed in [57] for DTs deployed in the vehicular context. The motion prediction time can be deployed as a separate container, running Python routines to execute the motion prediction algorithm. Measured inference times are in the order of milliseconds up to hundreds of milliseconds, in the worst case. Hence, the mobility can be predicted in an efficient manner, with not so much burden on the SDT side. An open source container orchestration engine, such as Kubernetes, can be leveraged for automating deployment, and managing containerized applications in a scalable manner, as proposed, for instance, in an SIoT-like context in [10], to support the instantiation and relocation of SDTs as decided by the MEC orchestrator.

All in all, both numerical results as well as the identified possible deployment options that match the conceived theoretical framework, suggest that the proposal can aim for scalability under different perspectives, even under large-scale SIoT implementations. Of course, there is room for improvement and for assessing the suggested implementation options.

## 5. Conclusions and future works

The proliferation of SIoT devices and their SDTs poses a challenge to orchestration procedures related to the placement and replacement of SDTs at the network edge. Fixed-interval relocation of SDTs may not be efficient and may lead to resource wastage. This work proposes a learning-based social-aware orchestration that predicts device mobility and makes judicious decisions about where and when to migrate SDTs efficiently. The proposed approach outperformed a fixed-interval approach for relocation and achieved a proximity of 99% to the ground truth in terms of service discovery latency.

However, several issues regarding SDT design remain to be addressed and deserve further investigation. They include, among others, the design of privacy-preserving solutions for position data retrieval [58]. More specifically, the collection and processing of position data for prediction purposes raises privacy concerns that require protection measures against potential attacks targeting learning models within SDTs. This implies the need to implement intrusion detection systems and secure communication protocols to identify network-based attacks, such as signature-based anomalies in network traffic patterns, and to ensure network security.

Regarding ML-assisted mobility prediction, DL-based prediction methods may deserve special attention when dealing with increased variability (both temporal and spatial) in motion patterns. In general, to improve the accuracy of the prediction, it may be appropriate to train the model on a large number of samples, which consider greater variability in the movement dynamics of SIoT devices. From this perspective, a possible emerging problem is related to the complexity of using a single model such as LSTM when dealing with long-time series data [59]. Similarly, combining LSTM with other neural networks can lead to a more complex and difficult-to-train structure [60]. In this

case, model transfer can be a useful approach, which allows training the model on a large number of samples and updating it with current data to provide predictions. This is faster and can be done in real-time because the model already has some weights, and training it from scratch is not needed.

## CRedit authorship contribution statement

**Olga Chukhno:** Writing – original draft, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Nadezhda Chukhno:** Writing – original draft, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Giuseppe Araniti:** Conceptualization. **Claudia Campolo:** Writing – review & editing, Supervision, Investigation, Conceptualization. **Antonio Iera:** Writing – review & editing, Supervision, Conceptualization. **Antonella Molinaro:** Writing – review & editing, Supervision, Conceptualization.

## Declaration of competing interest

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

## Data availability

Data will be made available on request.

## Acknowledgments

This work was supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on “Telecommunications of the Future” (PE00000001 - program “RESTART”).

## References

- [1] H. Kheddar, Y. Himeur, S. Atalla, W. Mansoor, An efficient model for horizontal slicing in 5G network using practical simulations, in: 2022 5th International Conference on Signal Processing and Information Security, ICSPIS, IEEE, 2022, pp. 158–163.
- [2] L. Atzori, A. Iera, G. Morabito, M. Nitti, The social internet of things (sIoT)—when social networks meet the internet of things: Concept, architecture and network characterization, *Comput. Netw.* 56 (16) (2012) 3594–3608.
- [3] L. Atzori, C. Campolo, A. Iera, G.M. Milotta, G. Morabito, S. Quattropiani, Sociocast: Design, implementation and experimentation of a new communication method for the internet of things, in: 2019 IEEE 5th World Forum on Internet of Things, WF-IoT, IEEE, 2019, pp. 662–667.
- [4] R. Minerva, G.M. Lee, N. Crespi, Digital twin in the IoT context: A survey on technical features, scenarios, and architectural models, *Proc. IEEE* 108 (10) (2020) 1785–1824.
- [5] A. Masaracchia, V. Sharma, B. Canberk, O.A. Dobre, T.Q. Duong, Digital twin for 6G: Taxonomy, research challenges, and the road ahead, *IEEE Open J. Commun. Soc.* 3 (2022) 2137–2150.
- [6] F. Guo, F.R. Yu, H. Zhang, X. Li, H. Ji, V.C. Leung, Enabling massive IoT toward 6G: A comprehensive survey, *IEEE Internet Things J.* 8 (15) (2021) 11891–11915.
- [7] L.U. Khan, W. Saad, D. Niyato, Z. Han, C.S. Hong, Digital-twin-enabled 6G: Vision, architectural trends, and future directions, *IEEE Commun. Mag.* 60 (1) (2022) 74–80.
- [8] O. Chukhno, N. Chukhno, G. Araniti, C. Campolo, A. Iera, A. Molinaro, Placement of social digital twins at the edge for beyond 5G IoT networks, *IEEE Internet Things J.* 9 (23) (2022) 23927–23940.
- [9] M. Amadeo, M. Martalò, G. Ruggeri, M. Nitti, Enabling social digital twins in the 6G era with information centric networking, *IEEE Commun. Mag.* (2023).
- [10] A. Lombardo, G. Morabito, S. Quattropiani, C. Ricci, Sociality-as-a-service: A new platform for networked digital twins, in: 2022 61st FITCE International Congress Future Telecommunications: Infrastructure and Sustainability, FITCE, IEEE, 2022, pp. 1–5.
- [11] O. Chukhno, N. Chukhno, G. Araniti, C. Campolo, A. Iera, A. Molinaro, Optimal placement of social digital twins in edge IoT networks, *Sensors* 20 (21) (2020) 6181.

- [12] C. Marche, L. Atzori, V. Pilloni, M. Nitti, How to exploit the social internet of things: Query generation model and device profiles' dataset, *Comput. Netw.* (2020) 107248.
- [13] A. Sedighi, S.M. Ulman, et al., Information presentation through a head-worn display ("smart glasses") has a smaller influence on the temporal structure of gait variability during dual-task gait compared to handheld displays (paper-based system and smartphone), *PLoS One* 13 (4) (2018) e0195106.
- [14] A. Choudhary, S. Gokhale, Urban real-world driving traffic emissions during interruption and congestion, *Transp. Res. D* 43 (2016) 59–70.
- [15] Q. Guo, F. Tang, T.K. Rodrigues, N. Kato, Five disruptive technologies in 6G to support digital twin networks, *IEEE Wirel. Commun.* (2023).
- [16] Q. Fan, N. Ansari, On cost aware cloudlet placement for mobile edge computing, *IEEE/CAA J. Autom. Sin.* 6 (4) (2019) 926–937.
- [17] Y. Lu, S. Maharjan, Y. Zhang, Adaptive edge association for wireless digital twin networks in 6G, *IEEE Internet Things J.* 8 (22) (2021) 16219–16230.
- [18] Z. Chen, W. Yi, A. Nallanathan, J. Chambers, Distributed digital twin migration in multi-tier computing systems, *IEEE J. Sel. Top. Sign. Proces.* (2024).
- [19] L.U. Khan, E. Mustafa, J. Shuja, F. Rehman, K. Bilal, Z. Han, C.S. Hong, Federated learning for digital twin-based vehicular networks: Architecture and challenges, *IEEE Wirel. Commun.* (2023).
- [20] L. Wang, L. Jiao, T. He, J. Li, M. Mühlhäuser, Service entity placement for social virtual reality applications in edge computing, in: *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, IEEE, 2018, pp. 468–476.
- [21] E. Carter, P. Adam, D. Tsakis, S. Shaw, R. Watson, P. Ryan, Enhancing pedestrian mobility in smart cities using big data, *J. Manage. Anal.* 7 (2) (2020) 173–188.
- [22] A. Kaltenbrunner, R. Meza, J. Grivolla, J. Codina, R. Banchs, Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system, *Pervasive Mob. Comput.* 6 (4) (2010) 455–466.
- [23] T. Bravenec, et al., UJI probes revisited: Deeper dive into the dataset of wi-fi probe requests, *IEEE J. Indoor Seamless Position. Navig.* (2023).
- [24] T. Bravenec, J. Torres-Sospedra, M. Gould, T. Fryza, UJI probes: Dataset of wi-fi probe requests, in: *2023 13th International Conference on Indoor Positioning and Indoor Navigation, IPIN, IEEE, 2023*, pp. 1–6.
- [25] O. Chukhno, O. Galinina, S. Andreev, A. Molinaro, A. Iera, Interplay of user behavior, communication, and computing in immersive reality 6G applications, *IEEE Commun. Mag.* 60 (12) (2022) 28–34.
- [26] S.Y. Han, M.-H. Tsou, E. Knaap, S. Rey, G. Cao, How do cities flow in an emergency? Tracing human mobility patterns during a natural disaster with big data and geospatial data science, *Urban Sci.* 3 (2) (2019) 51.
- [27] T. Fryza, T. Bravenec, Z. Kohl, Security and reliability of room occupancy detection using probe requests in smart buildings, in: *2023 33rd International Conference Radioelektronika, IEEE, 2023*, pp. 1–6.
- [28] J. Novák, L. Šýkora, A city in motion: Time-space activity and mobility patterns of suburban inhabitants and the structuration of the spatial organization of the Prague Metropolitan Area, *Geogr. Ann.: Ser. B Hum. Geogr.* 89 (2) (2007) 147–168.
- [29] M. Attaran, B.G. Celik, Digital twin: Benefits, use cases, challenges, and opportunities, *Decis. Anal. J.* (2023) 100165.
- [30] Y. Zhang, H. Zhang, Y. Lu, et al., Adaptive digital twin placement and transfer in wireless computing power network, *IEEE Internet Things J.* (2023).
- [31] J. Li, S. Guo, W. Liang, J. Wang, Q. Chen, Z. Xu, W. Xu, AoI-Aware user service satisfaction enhancement in digital twin-empowered edge computing, *IEEE/ACM Trans. Netw.* (2023).
- [32] Y. Zhang, W. Liang, Z. Xu, X. Jia, Mobility-aware service provisioning in edge computing via digital twin replica placements, *IEEE Trans. Mob. Comput.* (2024).
- [33] X. Chen, G. Han, Y. Bi, Z. Yuan, M.K. Marina, et al., Traffic prediction-assisted federated deep reinforcement learning for service migration in digital twins-enabled MEC networks, *IEEE J. Sel. Areas Commun.* (2023).
- [34] ETSI GS MEC 003 v3.1.1. Mobile edge computing (MEC); framework and reference architecture, 2022.
- [35] M. Picone, M. Mamei, F. Zambonelli, A flexible and modular architecture for edge digital twin: Implementation and evaluation, *ACM Trans. Internet Things* 4 (1) (2023) 1–32.
- [36] R. Klus, L. Klus, D. Solomitchii, J. Talvitie, M. Valkama, Deep learning-based cell-level and beam-level mobility management system, *Sensors* 20 (24) (2020) 7124.
- [37] D. Alekseeva, N. Stepanov, A. Veprev, A. Sharapova, E.S. Lohan, A. Ometov, Comparison of machine learning techniques applied to traffic prediction of real wireless network, *IEEE Access* 9 (2021) 159495–159514.
- [38] R. Klus, J. Talvitie, J. Vinogradova, J. Torsner, M. Valkama, Machine learning based NLOS radio positioning in beamforming networks, in: *2022 IEEE 23rd International Workshop on Signal Processing Advances in Wireless Communication, SPAWC, IEEE, 2022*, pp. 1–5.
- [39] R.P. Masini, M.C. Medeiros, E.F. Mendes, Machine learning advances for time series forecasting, *J. Econ. Surv.* 37 (1) (2023) 76–111.
- [40] Q. Liu, G. Chuai, J. Wang, J. Pan, Proactive mobility management with trajectory prediction based on virtual cells in ultra-dense networks, *IEEE Trans. Veh. Technol.* 69 (8) (2020) 8832–8842.
- [41] H. Gebrie, H. Farooq, A. Imran, What machine learning predictor performs best for mobility prediction in cellular networks? in: *2019 IEEE International Conference on Communications Workshops, ICC Workshops, IEEE, 2019*, pp. 1–6.
- [42] K.-L. Yap, Y.-W. Chong, Optimized access point selection with mobility prediction using hidden Markov model for wireless network, in: *2017 Ninth International Conference on Ubiquitous and Future Networks, ICUFN, IEEE, 2017*, pp. 38–42.
- [43] Z. Cheng, N. Chen, B. Liu, Z. Gao, L. Huang, X. Du, M. Guizani, Joint user association and resource allocation in HetNets based on user mobility prediction, *Comput. Netw.* 177 (2020) 107312.
- [44] M. Shabbir, S. Kandeepan, A. Al-Hourani, W. Rowe, LSTM based proactive access point selection and mobility load balancing for ultra-dense networks, in: *2024 International Conference on Artificial Intelligence in Information and Communication, ICAIC, IEEE, 2024*, pp. 452–458.
- [45] H.-S. Park, H. Kim, C. Lee, H. Lee, Mobility management paradigm shift: from reactive to proactive handover using AI/ML, *IEEE Netw.* (2024).
- [46] T. Bravenec, J. Torres-Sospedra, M. Gould, et al., What your wearable devices revealed about you and possibilities of non-cooperative 802.11 presence detection during your last IPIN visit, in: *2022 IEEE 12th International Conference on Indoor Positioning and Indoor Navigation, IPIN, IEEE, 2022*, pp. 1–7.
- [47] L. Klus, D. Quezada-Gaibor, J. Torres-Sospedra, E.S. Lohan, C. Granell, J. Nurmi, Towards accelerated localization performance across indoor positioning datasets, in: *2022 International Conference on Localization and GNSS, ICL-GNSS, IEEE, 2022*, pp. 1–7.
- [48] N. Chukhno, S. Trilles, J. Torres-Sospedra, A. Iera, G. Araniti, D2D-based cooperative positioning paradigm for future wireless systems: A survey, *IEEE Sens. J.* (2021).
- [49] A. Mei, J. Stefa, SWIM: A simple model to generate small mobile worlds, in: *IEEE INFOCOM, 2009*, pp. 2106–2113.
- [50] J. Perdomo, M. Ericsson, M. Nordberg, K. Andersson, User performance in a 5G multi-connectivity ultra-dense network city scenario, in: *2020 IEEE 45th Conference on Local Computer Networks, LCN, IEEE, 2020*, pp. 195–203.
- [51] F. Farina, D. Fontanelli, A. Garulli, A. Giannitrapani, D. Prattichizzo, Walking ahead: The headed social force model, *PLoS One* 12 (1) (2017) e0169734.
- [52] R. Landa, J.T. Araújo, R.G. Clegg, et al., The large-scale geography of internet round trip times, in: *2013 IFIP Networking Conference, IEEE, 2013*, pp. 1–9.
- [53] S.R. Gunn, et al., Support vector machines for classification and regression, *ISIS Tech. Rep.* 14 (1) (1998) 5–16.
- [54] F. Martínez, M.P. Frías, M.D. Pérez, A.J. Rivera, A methodology for applying k-nearest neighbor to time series forecasting, *Artif. Intell. Rev.* 52 (3) (2019) 2019–2037.
- [55] G. James, D. Witten, T. Hastie, R. Tibshirani, *An Introduction to Statistical Learning*, vol. 112, Springer, 2013.
- [56] L. Atzori, C. Campolo, B. Da, R. Girau, A. Iera, G. Morabito, S. Quattropiani, Smart devices in the social loops: Criteria and algorithms for the creation of the social links, *Future Gener. Comput. Syst.* 97 (2019) 327–339.
- [57] C. Campolo, G. Genovese, A. Molinaro, B. Pizzimenti, G. Ruggeri, D.M. Zappalà, An edge-based digital twin framework for connected and autonomous vehicles: Design and evaluation, *IEEE Access* (2024).
- [58] Y. Wang, Z. Su, S. Guo, M. Dai, T.H. Luan, Y. Liu, A survey on digital twins: Architecture, enabling technologies, security and privacy, and future prospects, *IEEE Internet Things J.* (2023).
- [59] S. Liu, I. Ni'mah, V. Menkovski, D.C. Mocu, M. Pechenizkiy, Efficient and effective training of sparse recurrent neural networks, *Neural Comput. Appl.* 33 (2021) 9625–9636.
- [60] A. Zhang, Z.C. Lipton, M. Li, A.J. Smola, *Dive into Deep Learning*, Cambridge University Press, 2023.