



UNIVERSITÀ DI PARMA

ARCHIVIO DELLA RICERCA

University of Parma Research Repository

Digital Twin for Continual Learning in Location Based Services

This is the peer reviewed version of the following article:

Original

Digital Twin for Continual Learning in Location Based Services / Lombardo, G.; Picone, M.; Mamei, M.; Mordonini, M.; Poggi, A.. - In: ENGINEERING APPLICATIONS OF ARTIFICIAL INTELLIGENCE. - ISSN 0952-1976. - 127:(2024). [10.1016/j.engappai.2023.107203]

Availability:

This version is available at: 11381/2977055 since: 2024-04-10T08:29:38Z

Publisher:

Elsevier

Published

DOI:10.1016/j.engappai.2023.107203

Terms of use:

Anyone can freely access the full text of works made available as "Open Access". Works made available

Publisher copyright

note finali coverpage

(Article begins on next page)

Digital Twin for Continual Learning in Location Based Services

Gianfranco Lombardo^{a,*}, Marco Picone^{b,**}, Marco Mamei^b, Monica Mordonini^a and Agostino Poggi^a

^aDepartment of Engineering and Architecture - University of Parma, Parma, Italy

^bDepartment of Sciences and Methods for Engineering (DISMI) - University of Modena and Reggio Emilia, Modena, Italy

ARTICLE INFO

Keywords:

Digital Twins
Location Based System
Intelligence
Internet of Things
MLOps
Deep Learning
Continual Learning

ABSTRACT

Decoupling the physical world and providing standardized service interfaces is still challenging when developing Location Based Services (LBS). This lack also hinders the possibility of developing Intelligent services on top of LBS architectures. In this paper, we propose a multi-layer Digital Twin-based architecture that aims to enable the development of machine learning-based Intelligent LBS (I-LBS) that are able to adapt, evolve, and perform Continual Learning (CL). The platform uses Digital Twins to ensure physical abstraction and provide cyber-physical knowledge to the I-LBSs, which is defined as an execution graph of operation modules. Finally, we simulated a use-case for this platform in the complex scenario of Healthcare organization and management where the I-LBS classifies allowed/not allowed trajectories of users inside a real-existing hospital scenario depending on their role in the organization. The use case is implemented as a Deep Learning-based reconstruction task of high-resolution trajectories processed by the DT architecture that also deploys the I-LBS. The platform is evaluated in terms of physical complexity and computational time on the DT side and on both a traditional machine learning setting and a replay-based CL one for the intelligence side to demonstrate the flexibility and adaptability features introduced by the components for dynamic or unseen scenarios.

1. Introduction

During the last decades, Indoor Positioning Systems (IPS) represented one of the main challenging industrial and research topics within the context of Location Based Services (LBS). The possibility to locate people or physical assets through a network of deployed sensing devices represents an appealing opportunity in multiple application scenarios where GPS (Global Positioning System) or other satellite technologies provides reduced precision fail (e.g., inside buildings and underground locations). At the same time, the Internet of Things (IoT) revolution brought a new wave of technologies, protocols, and architectural patterns that revitalized IPS and LBS applications through the use of distributed, connected, and intelligent Smart Objects (SOs).

Notwithstanding this evolution, the massive heterogeneity of physical devices and the presence of domain-specific vertical solutions (custom protocols and data formats) represent one of the main issues in the IoT ecosystem, directly affecting the adoption and exploitation of Indoor Location Based Services. Each vendor provides its own implementation in terms of APIs (Application Programming Interfaces), data structures, and architectural deployments ranging from cloud-only approaches to the possibility of running services also on edge.


This issue has a direct impact not only in terms of interoperability among devices and services but it is also limiting deployment longevity and modularity and the possibility of

designing intelligent and cognitive LBS applications on top of the physical layer without the concrete risk of keep reinventing the wheel and create isolated interesting solutions. This issue also hinders the adoption of the recent developments in Machine Learning (ML) and Deep Learning (DL) to the development of intelligent LBS, given the complexity of realizing such algorithms while taking into account the heterogeneity and variability of the underlying IoT system.

Recently, IoT revitalized the concept of Digital Twin (DT), originally introduced between 1999 and 2002 (Tao et al. (2019)) as a virtual clone of a physical object. A DT is able to map and replicate relevant properties and functionalities (all of them or a subset according to the context) and mutually cooperate and co-evolve with its physical counterpart through bidirectional interactions and communications. DTs represent an interesting tool for effectively decoupling the physical complexity and heterogeneity with the need for abstraction and uniformity of the digital layer. Digital Twins decouple the physical layer by offering a single uniform interface to data that enables the development of ML-based intelligent LBS. A key factor with models that learn from data is providing clean, accurate, consistent, and complete data as input. This aspect becomes even more challenging when data ingestion is performed starting from physical and heterogeneous contexts. Indeed, data sources can be different in terms of sampling rates, format, and features. As a consequence, a Digital Twin architecture that acts as a middle layer, providing abstraction and uniformity, may enable the development of intelligent services by leveraging, on one side, data and information coming directly from the environment, and on the other side, the machine learning techniques and paradigms currently available.

*G. Lombardo

**M. Picone

 gianfranco.lombardo@unipr.it (G. Lombardo);

picone.m@unimore.it (M. Picone); marco.mamei@unimore.it (M. Mamei); monica.mordonini@unipr.it (M. Mordonini); agostino.poggi@unipr.it (A. Poggi)

ORCID(s): 0000-0003-1808-4487 (G. Lombardo)

On the other hand, ML processes should be automated and operationalized to be brought into a production environment. Digital Twins represent a key factor in enabling and implementing the so-called Machine Learning Operations (MLOps) that aim to solve the issue of designing and maintaining productive ML models. MLOps aims to provide tools and functionalities to achieve Continuous integration (CI), Continuous delivery and deployment (CD), and model monitoring. Moreover, ML models require maintenance and updates to better fit changes in the environment and to improve their performances and thus need uniform access to features and data but also reusable execution pipelines for fast and automated re-deployment of Models (AutoML).

Moving further, these AutoML requirements become even more crucial when considering the Continual (Life-long) Learning (CL) paradigm: It provides solutions to learn continuously and adaptively about the external world and enables an autonomous incremental development when data and tasks become available only during the time (Parisi et al. (2019)). CL makes it possible to smoothly update the intelligent model to consider different tasks and data distributions unseen in the past training steps without losing the acquired useful knowledge. To achieve this goal in a real scenario, CL might benefit from a virtual abstraction that provides uniformity and transparency with respect to the physical layer and collects and prepares data depending on the needs of the intelligent model or service, enabling intelligent LBS as a service.

The Digital twin architecture we propose in this paper sets itself apart from existing solutions through its emphasis on flexibility and modularity. In contrast to existing solutions that depict DTs as passive components, lacking a broader ecosystemic vision and resulting in limited dynamic activity resembling a monolithic data repository, our design takes a different perspective. We view DTs as independent and active software components, facilitating an effective cyber-physical digitalization process and empowering the enhancement of capabilities offered by the deployed physical entities. Moreover, the decoupling of the physical world from the cyber-intelligence level is achieved by providing uniformity of data and AutoML capabilities in order to enable continual learning of intelligence-based new tasks whenever they emerge over time.

To validate and prove the benefits of our architecture, we focused on a use case that consists in an intelligent location-based service that aims to recognize normal and anomaly trajectories of users inside a real hospital ward. The intelligent service is based on a ML model that is trained with unlabelled data coming from a Digital Twin architecture, where unlabelled in this context means that the Digital Twin is unaware of anomaly trajectories but only knows what the role of the user that is performing the trajectory is. In order to deal with this context, we decided to leverage a self-supervised learning paradigm where the model learns how to reconstruct normal trajectories from trusted users, and it is then applied to a test set with normal and anomaly trajectories to prove its ability to recognize anomalies.

The main contributions of this paper are the following:

- A DT-based MLOps platform that handles and solves heterogeneity of the physical layer through the adoption of DTs aiming to develop intelligent LBS by exploiting cyber-physical knowledge.
- We show in a simulated hospital environment how this platform enables the development of an intelligent LBS that aims to automatically recognize anomaly trajectories of users, depending on their roles in the organization, by continually learning from their trajectories during the time.
- Finally, we evaluate the benefits introduced by the entire architecture, both in terms of system-level performance and application complexity and performance of the learning and anomaly detection tasks.

2. Related Works

The availability of novel technologies such as Mobile Computing and Internet of Things made LBS a fast-developing research field in the last years with several industrial and research product applications. However, several challenges are still considered open questions, especially at the core of LBS development: positioning, modelling, the analysis of LBS-generated data, as well as social, ethical, and behavioural issues (Jiang et al. (2021)). According to (Huang et al. (2018)), LBS still lacks standardization of the service interface, especially for indoor positioning solutions that yield different levels of accuracy and reliability, notwithstanding they rely on similar sensors, infrastructures, and positioning techniques. This latter is often related to different modelling of the indoor environments and the lack of use of metadata to characterize and specify the service interface with a transparent application layer able to effectively decouple the physical layer from the application layer. In most cases, indoor LBS applications often require more semantic than purely geometrical information and strongly depend on the indoor space model and on the particular use case (Afyouni et al. (2017)). In addition, data analysis and knowledge discovery from LBS-generated data are becoming fundamental for different applications in several research fields. However, the increasing amount and complexity of LBS-generated geospatial data require additional modelling and appropriate storage to provide analytics and to be exploited by the recent advantages of Machine Learning. Most of the literature on this topic focuses on location-based social media data with the main challenge of privacy-preserving (Gupta and Rao (2017); Jiang et al. (2021)). On the other hand, another research branch focuses on the application of ML techniques to LBSs to increase the accuracy, precision, and reliability of data from sensors by also exploiting Digital-Twin-based architectures Zhao et al. (2021).

The scientific and industrial communities recently worked to define and re-shape the role of Digital Twins and their

responsibilities among different application domains, particularly in relation to IoT-oriented use cases (Tao et al. (2019); Barricelli et al. (2019); Koulamas and Kalogeras (2018)). Some studies have also started to discuss general-purpose frameworks to enable cross-domain applications, but they mainly envision DTs as passive components (represented, for instance, as description documents (Autiosalo et al. (2021))) or isolated entities (lacking the vision of DTs as an ecosystem (Eramo et al. (2021)).) Ricci et al. (2021) introduces an ecosystem and Web-oriented point of view: in particular, its authors propose a pervasive and distributed abstract model for DT environments. In addition, for the industrial application domain, an appealing perspective has been introduced by Radanliev et al. (2022) by proposing a new approach to design DTs of physical devices/processes represented through conceptual diagrams. Furthermore, the recent combination between DTs and edge computing technologies opened the possibility of deploying and executing twins as close as possible to physical assets in order to enable new real-time cyber-physical interactions (Bellavista et al. (2021); Picone et al. (2021b)). In this evolving and revitalized Digital Twin ecosystem, the experimentation and integration of DTs within the context of LBS application is at an early stage, and the possibility of exploiting them as an effective enabler for intelligent location services is still under exploration.

In addition, DTs may play several roles in filling the current gap between the physical world and MLOps principles with several benefits (Fujii et al. (2021)). Indeed, the MLOps field has emerged as a discipline to help bridge the gap between the development of ML models and their operations with several principles and technical components (Kolltveit and Li (2022)), most of which are related to Data management and abstraction for ML models auto-deployment and update when external factors occur or when performance degrades because of external changes.

According to Kreuzberger et al. (2022), an MLOps architecture should guarantee at least the following functionalities: (i) *Continuous integration, delivery and deployment (CI/CD)*, (ii) *Workflow orchestration* to coordinate the tasks of an ML workflow pipeline according to directed acyclic graphs that define the execution order by considering relationships and dependencies, (iii) *Reproducibility and traceability* by storing the necessary metadata (such as hyper-parameters and performance), (iv) *Continuous ML training and evaluation* when more recent data are available, (v) *Continuous Monitoring* to evaluate if performance degrades during the time and if any update or re-training is necessary.

The MLOps landscape is quickly growing with several tools and platforms that aim to implement the main components for MLOps (Matsui and Goya (2022)) notwithstanding several open problems related to modelling how MLOps pipelines should deal with real-time streaming of information coming from the physical world such as the case of LBS or in general at the edge level (Leroux et al. (2022)).

We believe Digital Twins may provide the necessary standardization and uniformity to solve most of the open

problems in LBS. In light of this, Location-based services Digital twin-based can offer the opportunity of developing intelligent services that exploit Machine Learning, not only to achieve better performance but also to learn directly from LBS-generated data how to offer novel and smarter services. To the best of our knowledge, this promising development has not been already deeper explored in literature.

Our work aims to propose a general framework that combines all the recent developments in the LBS, Digital Twin, and Machine Learning fields. In particular, we took into account the latest developments in the Deep Learning literature related to Continual Learning methodologies.

The central challenge of Continual Learning generally arises from the sequential nature of learning: when learning the k_{th} task from a dataset D_t , the old training sets D_1, \dots, D_{k-1} of the previously learned tasks may be inaccessible for several reasons (e.g. privacy-preserving, storage costs reduction)(Wang et al. (2023)). Therefore, it is critical to capture the probability distributions of both old and new tasks in a balanced manner, ensuring the so-called stability-plasticity trade-off (Lee et al. (2020)), where excessive learning plasticity (learning new tasks with fewer regularization mechanisms) or memory stability (giving more importance to previously learned distribution) can largely compromise each other (Lee et al. (2017)). Indeed, traditional deep-learning models show a well-known forgetting mechanism (Catastrophic Forgetting -Tadros et al. (2022)) when learning different tasks sequentially that can currently alleviate by leveraging on the hyper-parameters considered in the stability-plasticity trade-off (Kemker et al. (2018)).

Continual Learning techniques try to solve this issue by leveraging algorithms that alleviate Catastrophic Forgetting. It is possible to distinguish three categories of CL techniques: (1) *Replay-based methods*: When learning a new task, the model is exposed with a variable percentage to examples from the previously learned tasks (Buzzega et al. (2020); Rolnick et al. (2019)). (2) *Regularization-based methods*: Continual learning is performed by acting on the parameters of the model by discouraging the updating of the neural network's layers that are deemed relevant for the single tasks. Unfortunately, these methods do not scale well when increasing the number of tasks (Zenke et al. (2017)). (3) *Architectural methods*: Developing ad-hoc solutions where different parts of the model take care of each task exclusively (Schwarz et al. (2018); Lombardo et al. (2022b))

Similar approaches can also be considered in the more general context of Online learning, where the goal is learning better a single task when new information and data are available (Maltoni and Lomonaco (2019)). In Hashash et al. (2022), the authors propose a continual learning approach to preserve synchronization between real and digital entities in a continuously growing digital twin eco-system.

To the best of our knowledge, our research work is the first one where a Digital Twin architecture exploits a Continual Learning framework to develop intelligent location-based services. In particular, we show how Digital Twins

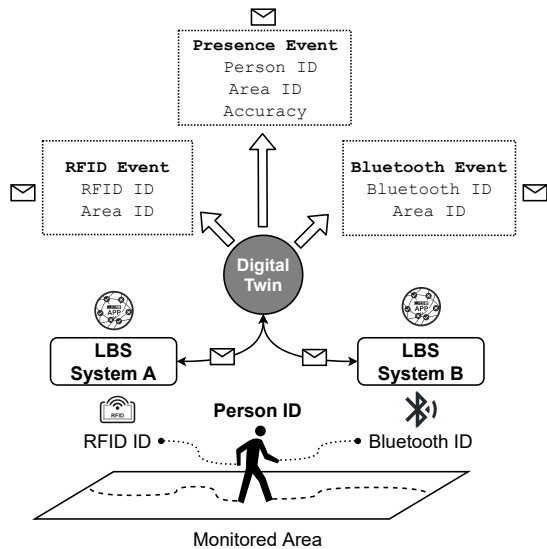


Figure 1: An example of the DT's disambiguation is where a tracked user is perceived by multiple heterogeneous technologies in the same area. The DT is responsible for unifying the representation of the received data and generating a new abstract reference of the presence merging available information.

supports the development of complex Continual Learning mechanisms in LBS by hiding the complexity of dealing with the underlying IoT system.

3. Cyber-Physical Intelligence Architecture

Digital Twins are digitalized software replicas of physical objects with the responsibility to clone available resources and functionalities and to extend existing behaviors with new capabilities. The state-of-the-art (Minerva et al. (2020); Minerva and Crespi (2021)) identifies a set of core properties for Digital Twins (DTs). Among them, *reflection* and *augmentation* are two of their most important functionalities allowing to create and keep the digital replica synchronized with the target properties and functionalities and extending the list of provided features and behaviors.

The architectural contribution and aim of the paper are to propose a Digital Twin-driven abstraction to simplify the definition of cross-domain intelligent and scalable I-LBS services. DTs represent the perfect tool to effectively merge the physical and the cyber worlds by handling as close as possible to the assets the heterogeneity in terms of platforms, communication protocols, and data formats in order to provide a uniform abstraction and service level functional to upper services and layers.

As previously anticipated, a shared and homogeneous abstraction is required in order to properly decouple functionalities between the management of localization devices and data and the provided services by avoiding the creation of domain-specific silos and strong bounds between a high-level functionality (e.g., detect anomaly movements in a target area) and a specific target deployment. For example, as

depicted in Figure 1, two localization platforms can generate events through heterogeneous technologies (e.g., Bluetooth and RFID) associated with the same person moving in a monitored area. In this scenario, we can introduce an intermediate cyber-physical entity (e.g., a DT) handling the disambiguation process and managing incoming data from various tracking technologies within a given location. It can also encapsulate the task of standardizing the representation of the collected data and creating a new user's representation by combining all the available information simplifying algorithms and business logic of upper intelligent layers.

Intelligent services should be aware only of the information required to execute their algorithms and to provide the designed services without handling or even knowing the underlying complexity. In a traditional architecture, the two streams of information are collected by independent and proprietary subsystems. Their effective integration is possible only by introducing an additional third-party module to disambiguate the two data streams and generate a new augmented event. On the opposite, with the proposed approach, this complexity is directly managed by DTs responsible for the target area managing the local physical complexity and augmenting directly provided capabilities through the introduction of intelligent services learning and operating on top of a uniform digitalized layer.

3.1. Multi-Layer Blueprint Architecture

In order to achieve that challenging goal we envisioned a new multi-layer architectural blueprint schematically depicted in Figure 2. The proposed approach is schematically composed of a hierarchical structure of DTs sharing the same high-level cyber-physical structure but with different responsibilities of managing real-time events originating from the physical assets and coordinating ML operations such as model training and execution. Envisioned DTs, build a homogenous and smart layer on top of the heterogeneity and fragmentation of the physical world to support users, system administrators, and external digital services to effectively exploit I-LBS solutions. The aim of the proposed solution is on the one hand, to decouple the physical heterogeneity of connected LBSs and on the other hand to simplify the design, implementation, and execution of intelligent location-driven algorithms through the use of hierarchical, distributed, and interconnected twins. The proposed design stands out from existing approaches, considering both research contributions Autiosalo et al. (2021); Eramo et al. (2021) and production-ready solutions¹², in terms of flexibility and modularity. Unlike other solutions, which tend to adopt a monolithic perspective where a single centralized software entity manages the complexity of physical objects and DTs. Furthermore, existing approaches portray DTs as passive components, typically represented through description documents, or as isolated components without considering the possibility of a broader DT ecosystem. Consequently, these frameworks

¹Azure Digital Twins: <https://azure.microsoft.com/en-us/products/digital-twins/>

²Eclipse Ditto: <https://www.eclipse.org/ditto/>

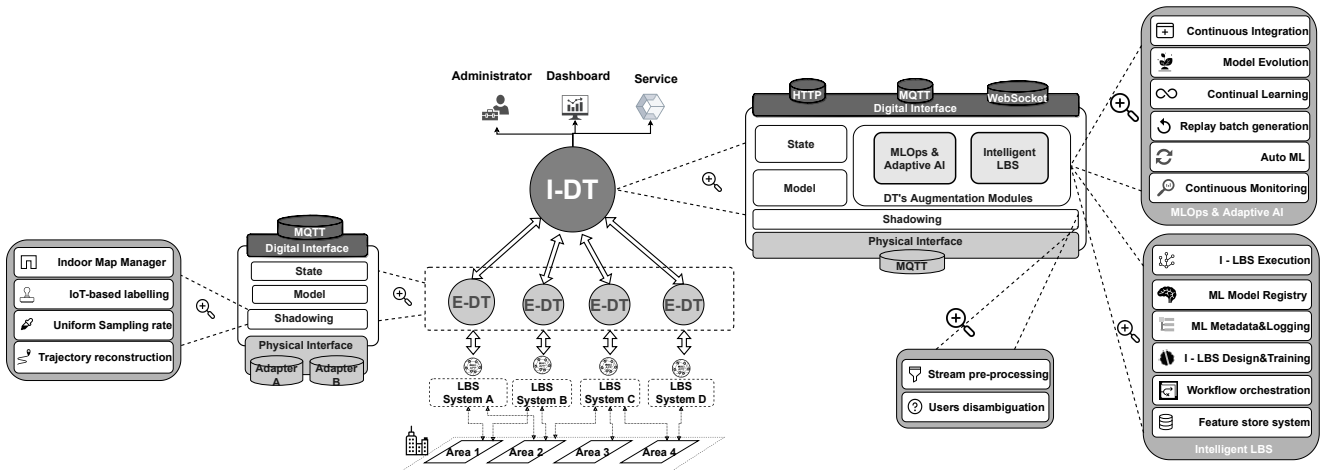


Figure 2: The schematic representation of the proposed approach where a hierarchical structure of DTs is responsible for handling real-time events from physical monitoring platforms enabling intelligent services to operate without handling the complexity of the physical deployment.

lack dynamic activity and resemble more of a data repository rather than active DTs with an internal model and dedicated behavior. Our design investigates a different and more flexible approach where DTs are modeled, implemented, and designed as independent and active software components in charge of digitalizing a specific physical twin with also the possibility to augment its capabilities. Involved DTs share a common internal structure characterized by the following primary components:

- *Physical Interface*: responsible for the communication with physical objects through multiple protocols mapped into sub-modules and adapters. Generates events from the physical world useful for the DT's model and the shadowing component to enable the digitalization process and receives actions or triggers that should be executed on the device.
- *Digital Interface*: exposes the status of the twin together with its capabilities and behaviors to external digital services adopting different protocols at the same time through multiple configurable adapters and connectors.
- *Shadowing*: the core function responsible to interact with the physical world through the *Physical Interface* and collecting relevant data for the *Model* and forwarding actions received from the digital world.
- *State*: defines the twin in terms of properties, events, actions, and relationships associated with the twin according to its model and information coming from the *Physical Interface*.
- *Model*: a set of functional modules responsible to shape the behavior of the DT and how it can create a digital replica of a physical entity with respect to its context and the designed application goals. It interacts

with the *Physical Interface* and the *Shadowing* component to handle physical variations and with the *State* to update the DT's representation.

- *Augmentation Functions*: optional and additional functions allowing the DT to extend its capabilities with respect to those that are originally associated with the bound physical objects through the interaction with the DT's *Model* and *State*.

Relying on this common modeling, we defined two classes of DTs denoted as **Environment DT (E-DT)** operating close to the physical layer and **Intelligence DT (I-DT)** at the higher hierarchy to provide intelligent services. Each of them has specific architectural responsibility and with the aim to effectively decouple the complexity of managing an I-LBS cyber-physical system. At the bottom layer, we have one or multiple E-DTs responsible to manage a target area of interest according to the designed abstraction granularity (e.g., an entire building or a specific floor) and the application goal (e.g., access control, path planning etc ...). Each E-DT can be configured to communicate through multiple protocols (e.g., HTTP, MQTT) and interaction patterns (e.g., RESTful, Pub/Sub, Push, Polling etc ...) with one or more LBS systems providing information belonging to the same area of interest in order to receive live updates about the tracked moving devices. The same physical area can be covered by multiple localization technologies at the same time and each E-DT is also in charge of building a uniform abstraction of the target space and moving entities by translating fragmented information coming from LBS sub-systems into a uniform representation (as illustrated in Figure 3) and directly handling the disambiguation among multiple representations of the same asset (e.g., a doctor moving through hospital rooms tracked at the same time by different platforms). The main responsibility of these E-DTs is to map and homogenize the fragmentation of the physical through its shadowing module in order to provide a uniform

digital representation to the upper layer and to support data analysis, learning and actionability. It processes information coming from deployed LBS sub-system with the final goal of processing and extracting high-level knowledge coming from the external environment. It is the first data layer that aims to prepare data in a suitable format to feed DTs higher in the hierarchy.

The combination of multiple E-DTs contributes to build and provide to the overall platform with prior knowledge of the map environment but they can also perform shallow indoor mapping by analyzing batches of trajectories coming from the field data. Finally, they provide a uniform layer for the trajectories that are transformed and pre-processed to have a uniform sampling rate depending on the function requirements coming from the sensing device and for reconstructing the path followed by each tracked user. Moreover, it provides all the operations related to feature scaling, dimensionality reduction, and feature selection. In this way, the next DT can operate directly with the desired data format in a transparent way with respect to the physical environment. In light of this, the E-DTs do not include ML pipelines, but they only operate to prepare data for the next layer of intelligent DTs.

On top of this layer, we have a composed I-DT responsible for collecting data from the E-DTs through the use of a uniform data structure and format and a unique standard protocol like MQTT. The shadowing function of these digital twins directly works on homogeneous data with the specific duty of providing users disambiguation when the same user is detected by multiple technologies in the same area. Distinguishing users on the basis of the technology that performed the detection and their disambiguation is also useful for performing simple trajectories' labelling that can be used, for example, to distinguish different categories of users in the environment and their roles. However, the core responsibility of the I-DT is to introduce in the architecture augmented capabilities for supporting intelligent behaviours through dedicated I-LBS ML techniques and for managing its life cycle and adapting to physical variation through MLOps and adaptive AI solutions. Each of these complex functionality has been mapped into a dedicated augmentation module integrated into the DT with the aim to build a self-contained and augmented digitalized entity of a physical location providing an intelligent entry-point for observing and interacting with digital applications unaware of the complexity of the provided features and their management. Introduced augmentation modules and their capabilities have been detailed in the following sections.

4. Cyber-Physical Intelligence

Cyber-physical intelligence is achieved by leveraging the Augmentation modules in the I-DT (Figure 2). In particular: (i) the Intelligent LBS which provides the functionalities related to the Continuous integration, deployment, and delivery of the Intelligent Location-based Service; (ii) The MLOps module which plays the role of supervisor of all the

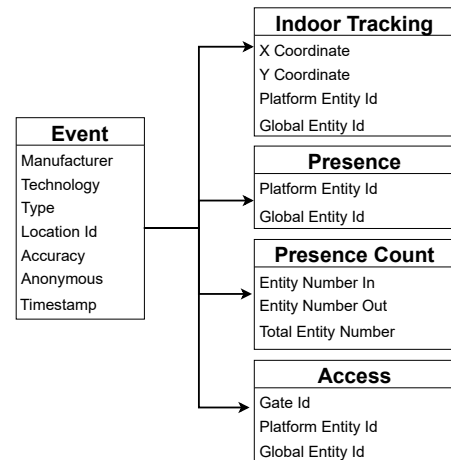


Figure 3: An example of the uniform data structure adopted in the proposed solution to map location-based events into a uniform representation.

ML models and provides functionalities for redeployment and continual learning when performance degrades, when there are changes in the environment or when new tasks emerge that need to be learned.

4.1. Intelligent LBS

This model creates intelligent location-based services combining multiple ML tasks in a complete execution graph. The execution graph defines the workflow (Training, Inference, Adaption, Continual Learning) that creates the services. An example of general Workflow orchestration is provided in Figure 4. The I-LBS is, therefore a directed graph that: a) defines the behavior and execution of the location-based service; b) promotes re-usability and sharing of intelligent nodes across different services.

This execution-graph design architecture offers several advantages: First, it simplifies the modeling of complex tasks that may require different steps and parallel pipelines to carry out the result. In addition, it promotes the re-usability of each component. This last point, although best development practice, is also crucial when the goal is performing Continual Learning. Indeed, the I-LBS, seen as an execution of reusable modules, enables to implement the execution of the steps that should be performed to adapt a model to the novelties coming from the physical world and which needs to learn new tasks from data that become available during the time. This is clearly possible, in particular thanks to the uniformity over the heterogeneity that is guaranteed by the E-DT layer, but also because training and fine-tuning of the models can be automatically performed (AutoML) by running a sequence of intelligent modules when requested without any knowledge of the previous history except for the required data format and the pre-trained model retrieved from the register.

In terms of MLOps design, this sub-part of the Digital Twin also acts as Model Serving Component (Kumara et al. (2022)) for online inference for real-time prediction on new data. It also includes a reference to the ML Model Registry,

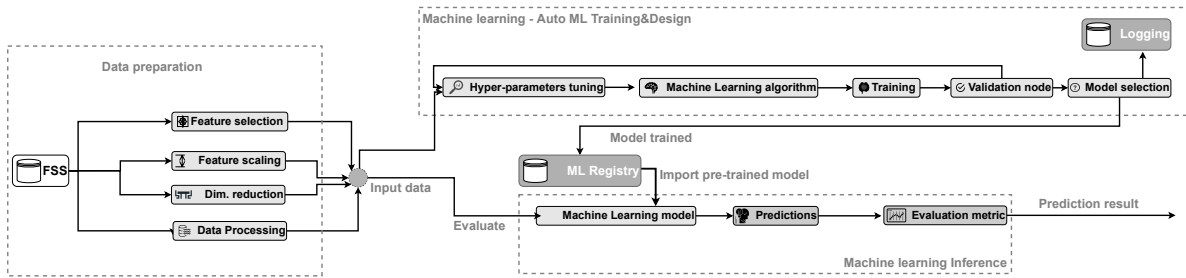


Figure 4: Generic Workflow orchestration to model and implement an I-LBS. Different tasks can be run in parallel, starting from the same data and following different execution walks across the graph.

a central data layer to store the current and most recent ML model together with its metadata (training time, best hyper-parameters identified on the validation set at the evaluation step). Metadata and logging storage are fundamental to ensure model versioning for reproducibility and traceability when the model should be updated and evaluated in terms of performance.

Finally, it implements the Feature Store System (FSS), which plays an important role in abstracting and exchanging cyber-physical knowledge to the services. The FSS is a data management system that manages and serves features to machine learning models, especially in the case of AutoML for automatic re-training of the ML models and to build different models that share subsets of features. FSS has the duty of building features from cyber-physical information, combining features into training data, calculating and serving features in production, monitoring features in production, and achieving consistency between training and serving data Kreuzberger et al. (2022). A feature store understands the data needed by a model, either for model training or to serve a prediction, as well as how that data connects to other data and the transformations required to get it into the right format. The Cyber-physical Knowledge DT is the main data source of the FSS, while the I-LBS module mainly acts as a consumer of features.

4.2. MLOps & Adaptive AI

This module is related to Adaptive Artificial Intelligence, which is still poorly investigated in MLOps pipelines. The DT periodically monitors the ML models' performance and queries the I-DT for the redeployment of pipelines.

It also enables the models to incremental training using more iterations and recovering from failures during the training. Furthermore, it is the main actor that reacts in case of changes in the environment to update the models and perform model evolution when more recent data are available. However, an intelligent service does not only need updates depending on environmental changes (e.g., a new organization of the environment, novel spaces, or more data for learning) but also because novel ML tasks emerge or are necessary for the organization (e.g., a novel category of users should be recognized).

In this last case, the module provides tools and data to perform Continual Learning by evolving the model structure or by querying a new training step with Replay-batch (Buzzege et al. (2020)). Thus, the module benefits from the Feature store system to extract batch of samples of the previously learned tasks, but can also support their synthetic generation when past data are not available anymore for several possible reasons (e.g., in health-care privacy preserving policies often permits to maintain data only for a certain amount of time). For this last scenario, the Digital Twin can train additional generative models once a new task is learned for the I-LBS. More details about Continual Learning modelling for the I-LBS are provided in Section 5.

Figure 5 shows an example of Workflow orchestration to perform Continual Learning using replay-based methods in the case of an I-LBS that initially leverages only a ML model to perform a task T_i but over time needs to be updated to also perform a new task T_{i+1} (e.g., consider a new class for a classification task). The new task T_{i+1} can be learned using a replay method without training the entire model for all the previous tasks already learned. Replay methods can be summarized as exposing the ML pre-trained model on some batches of previous tasks' examples to avoid Catastrophic Forgetting when learning a novel task. The architecture can retrieve a random batch as Replay for each task when past data are available or generate examples for task T_i while learning how to perform that task (Generative Replay Shin et al. (2017)). When the new task T_{i+1} should be learned, the model trained until task T_i is retrieved from the ML Register, and a replay batch for the previous tasks is generated with Generative Replay or directly queried to the ML Register. We will further highlight this concept when discussing our experimental case in section 5.

In summary the proposed DT-based architecture supports and simplifies the interaction with sensors and devices, and provides intelligent components supporting and enabling advanced I-LBS application like the one presented in the following section.

5. Experimental Use-case

In this section, we present the use case we exploited to test and validate our architecture. The main goal of the proposed I-LBS is to classify allowed/not allowed trajectories of

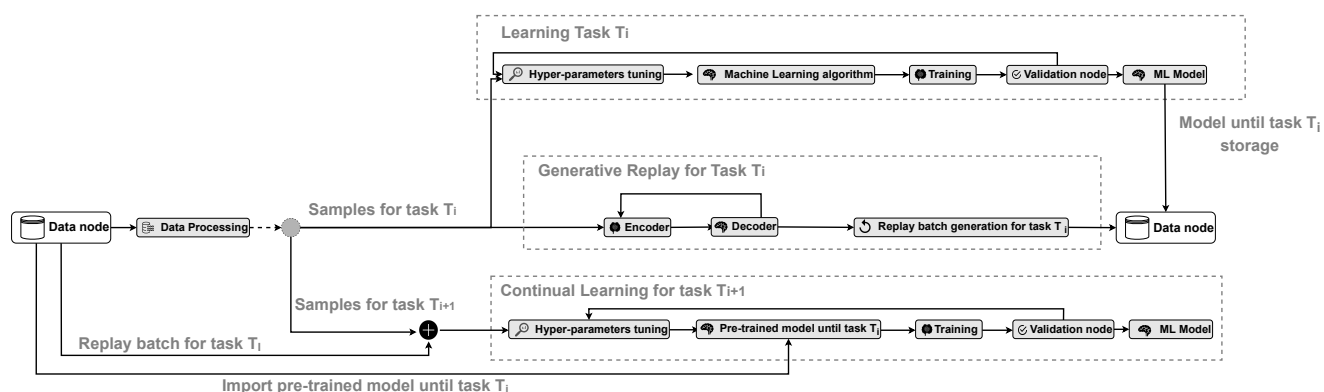


Figure 5: Workflow orchestration for a generic Continual Learning model exploiting Replay-based methods. The first task T_i is learned by training a traditional ML model using the same orchestration of Figure 4. At the same time, another set of nodes has the duty of learning a Generative model that learns how to generate examples for the task T_i on demand if the platform cannot store the original examples for this task. Finally, the same pre-trained model is updated to learn the new task T_{i+1} by leveraging its samples retrieved by the data node and a Replay batch of each previously learned task.

users inside a real-existing hospital scenario, depending on their role in the organization. Users can be common patients, maintenance staff, or medical staff and are located by exploiting different IoT technologies and protocols with fixed anchors that locate the user. Locating users and identifying and classifying their trajectories can be useful for multiple scopes, for example:

- For security reasons: Users are monitored to detect possible dangerous behaviours and avoid access into not allowed sections of the environment.
- Measuring how changes in the spatial organization of an environment can affect the efficiency of users' trajectories or understanding if any change in the organization induces anomaly trajectories. Management can have direct feedback about the efficiency of the hospital's planned routes whenever they are changed or detect critical points that need more attention.
- Measuring the time spent around a place for each user's category. This knowledge can be used to develop more complex predictive tasks: such as estimation of waiting times and predicting how long someone will occupy a region of the space or a service facility.

In the proposed use case, we deal only with the detection and classification over the time of allowed/not allowed trajectories, but the same Digital Twin architecture can be used to design I-LBSs for the other example we reported, such as one of our ongoing research project in healthcare³ where we are currently exploiting the architecture to learn how to estimate intra-operative times in the surgical block for each patient.

³"Development of a Machine Learning tool for process optimization in the healthcare context" - FIL-Quota Incentivante, University of Parma, Italy

Figure 7 shows a map segmentation of the real-existing hospital ward we used for the experiments. The environment is divided according to the main operations that are performed daily. Patients enter the ward by the waiting room, can proceed through the back office and reception area, and then visit the clinic areas (orange) or the areas where treatments are provided (light pink). They cannot access the maintenance area (violet) and the private spaces of clinicians, and the archive. On the other hand, the medical staff can access all the environments except the maintenance area and can enter the ward from multiple gates. Finally, the maintenance staff can enter from multiple gates, they can access the private area for maintenance, but we supposed they cannot access the archive and the private medical offices.

Each class of users is indoor-located by different technologies with different sampling frequencies and accuracy. This last information is next exploited to recognize each user's role and permissions. In our experimental modeling and implementation, we have successfully incorporated the support for two leading commercial LBS platforms renowned for their adoption at the same time of both WiFi and Bluetooth indoor localization capabilities. Specifically, we seamlessly integrated in the emulated environment the support for Cisco Meraki⁴ and Extreme Networks⁵, which are widely recognized and established platforms in this domain. Each platform exhibits distinct characteristics in terms of the update frequency based on the adopted tracking technology and both platforms have been utilized and associated with the possibility to track the different identified user types within the designed experiments. The environment DTs of our architecture provide physical abstraction and data representation uniformity. Moreover, they label the trajectories depending on the IoT technologies that have detected the users (Cyber-Physical Enrichment). They also compute

⁴Cisco Meraki Location Analytics - https://documentation.meraki.com/MR/Monitoring_and_Reporting/Location_Analytics

⁵Extreme Networks Location Analytics - <https://www.extremenetworks.com/product/extremelocation/>

basic indoor mapping of the environment when collecting new batches of users' trajectories during the time. This indoor mapping is subsequently used by the higher-hierarchy I-LBS Digital twin to perform user disambiguation and to detect changes in the environment's organization. Finally, they ensure a uniform sampling rate for the trajectories depending on the physical interface available.

Therefore the first group of DTs in the architecture provides a physical abstraction to develop an intelligent service based on unsupervised machine learning techniques. Indeed, in our use case, the machine learning model has no prior knowledge about the environment and the allowed trajectories for each category of users. Still, it learns the allowed ones in the environment for each category as an anomaly detection task over spatial location sequences and exploits the information about the technology used to track the user coming from the Cyber-Physical Enrichment.

The machine learning model learns to reconstruct the trajectory of each user using an encoder-decoder architecture aiming to reduce the Root Mean Squared Error (RMSE) between the input sequence and the reconstructed one. This is a common methodology used to detect anomalies in sequences and time series because the model fails to reconstruct sequences that rely on distributions that have not been seen during the training phase with a higher RMSE. We also provided the user category (Cyber-Physical Enrichment) as input for the model, together with its trajectory.

This approach provides three main advantages:

- Areas that are allowed to each category are learned by the model while training it with data coming from the usual trajectories of each category and avoid the need for supervised data and manual definition of different areas in the environment.
- This system can be directly applied in other contexts and hospitals by only collecting daily data to train the model for a new environment without the need for human intervention
- Adding new data sources or localization technologies and data sources, the architecture can perform both continual learning (learning novel categories) and online learning (updating the knowledge with novel emerging patterns of the users)

Figure 6 shows how the coordination and deployment of the E-DTs and I-DT, over time in our use-case and the main events that characterize their executions. The first phase regards the environment perception, where the generic users' trajectories are collected. The Environment DT process the trajectory coming from the physical interface, and associates a label according to the Cyber-Physical knowledge acquired that, in our use case, corresponds to associating the trajectory to a specific user category. After that, it samples the new trajectory in order to have a uniform length as input for the subsequent ML model. If the new trajectory includes unknown spatial points, the map knowledge of the environment is also updated. After that, the second DT named

I-DT begins to process the trajectory for the intelligent location-based service. It performs users' disambiguation to understand if the same user has been tracked with different technologies and define what is the final user category to be associated. Moreover, with the updated map knowledge and the trajectory, it updates the FSS and it transforms the trajectory according to the features available by also performing the common ML processing steps: feature scaling, dimensionality reduction, and outliers detection. After this step, the trajectory is ready to be fed in a machine learning model and added to the currently available dataset if it should be used as a training example or directly fed to the I-LBS if the trajectory should be classified as an anomaly or not in inference.

In the beginning, the I-DT also runs all the software components defined in the execution graph to train the first model. This operation is automatically performed once a batch of ready trajectories of trusted users is collected. At run time, the I-DT runs a different execution graph depending on the need of the service (Continual learning to update the model or only inference). Finally, the I-DT ensures that the trained model is stored in the ML Model registry. All the hyper-parameters and logging information are also stored in the architecture data layer.

5.1. Simulated environment

In order to develop and test this use case we built a virtual environment able to model and simulate the common user trajectories in the real hospital environment. We also modelled the data collection process by simulating that the locations visited by the members of each class of users are collected with a different sampling rate and with random noise. In this way, we simulate the localization of each class of users using a different IoT technology with different accuracy and precision. To achieve this goal, we used Agent-Based Modeling and Simulation techniques (ABMS) based on the Actor model Agha (1986). Each agent represents an individual who, depending on his role (patient, medical staff, maintenance), follows a different trajectory visiting different locations according to his permissions. The actors are implemented as concurrent processes using the ActoDeS framework Bergenti et al. (2014). We exploited this distributed architecture to generate data for both training and test-set. We simulate correct trajectories for training the model by exploiting the Boid model Angiani et al. (2018) to take advantage of an extended set of boid rules that allow the agents to reach the next location by randomly following either other agents or a set of alternative allowed paths. On the other hand, to create a test set that also includes negative samples (not-allowed paths for each category), we simulate random and partially wrong trajectories for each category in the same environment Lombardo et al. (2022a). Each trajectory has a different length, and the individuals can visit multiple locations.

Furthermore, as previously anticipated in Section 5 and with the aim to increase the realism of the emulated environment, we implemented the emulation and support for two

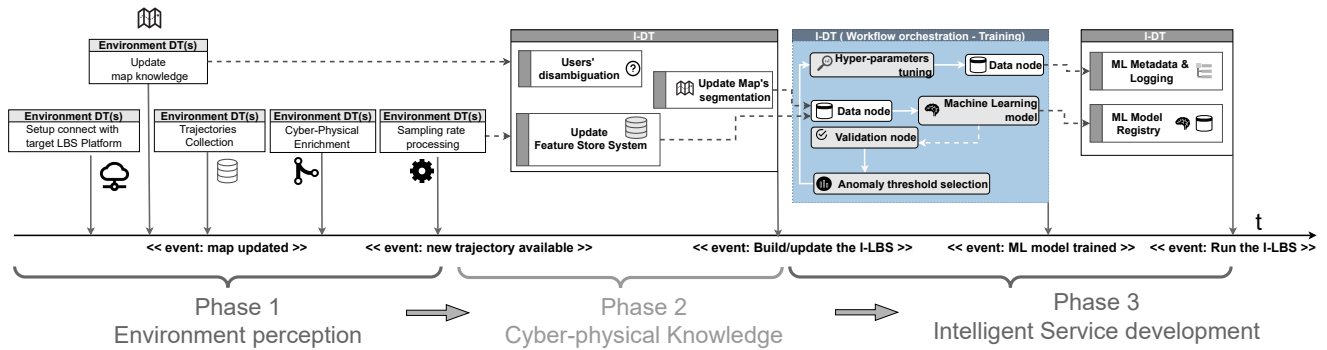


Figure 6: Sequence diagram that represents coordination over the time between the Environment DT and the I-DT for the anomaly detection task in the hospital and the main deployment steps such as the Workflow Orchestration with the intelligent nodes during the training phase performed by the I-DT.

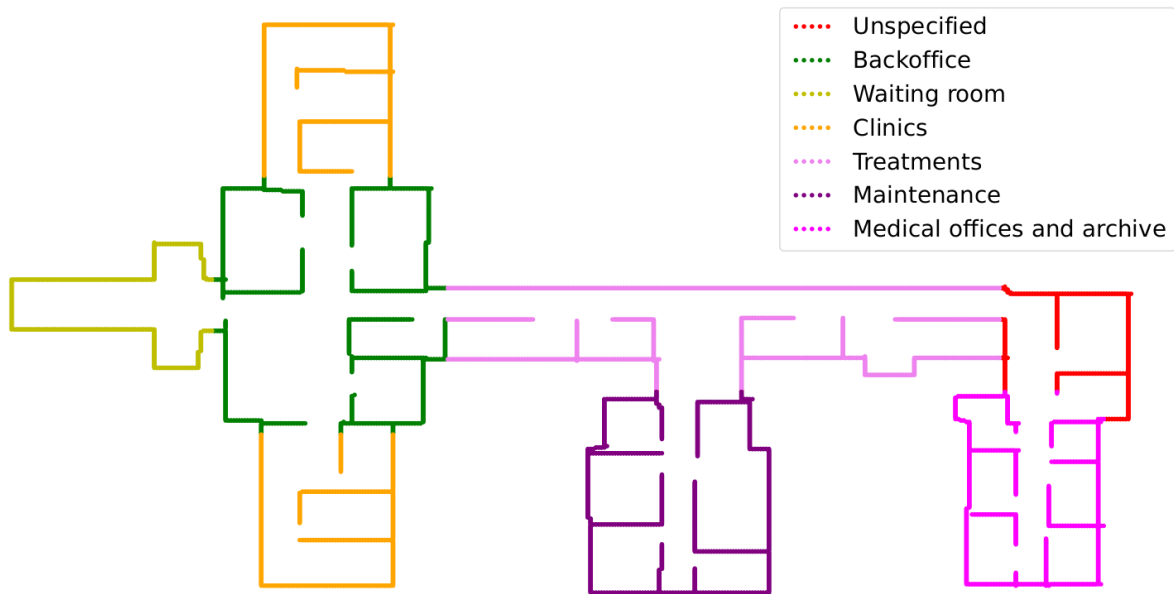


Figure 7: The real-hospital ward map used for the experimental part. The map is color-based and segmented according to the daily activities that are led in its environment.

of the main existing commercial LBS platforms supporting both WiFi and Bluetooth indoor localization respectively Cisco Meraki and Extreme Network. Each platform uses a different proprietary data format serialized through JSON and exposed with a custom HTTP communication pattern making each solution a siloed environment deployed of the cloud infrastructure of the different providers. The designed virtualized environment allows the generation of configurable indoor positioning information with respect to the target scenario and the associated map following platform-specific APIs and data structures with the goal of obtaining heterogeneous physical layers where multiple platforms operate at the same time in the same areas of interest.

5.2. The Anomaly detection task

The I-DT implements the intelligent service of recognizing allowed/not allowed trajectories for each user as an unsupervised anomaly detection task where the ML model learns

the underlying distribution of trusted users' trajectories. In particular, we exploited an encoder-decoder architecture based on Long-Short Term Memory (LSTM) deep neural networks (Figure 8). LSTM networks have been selected because of their ability to learn patterns on sequence data by considering and preserving temporal dependencies Yu et al. (2019). The anomaly detection model is trained in an unsupervised way by minimizing a mean squared error loss function between the input sequence (user trajectory) and the one generated by the Decoder without any supervised information related to anomalies of the trajectories. A trajectory is considered an anomaly when the Root Mean Squared Error (RMSE) of the two sequences is above a certain threshold that is automatically estimated by the DT using a validation set. According to the RMSE, the model classifies a trajectory for a user as not allowed when it results to be an anomaly and allowed otherwise.

The encoder network queries the FSS module to get the processed trajectory in the form of a sequence of 150 visited locations (x,y) along with the categorical variable that identifies the user category. After that, the model learns a compressed internal representation that is then used by the decoder network to reconstruct the trajectory. The two networks are trained with the Back-propagation algorithm and may be asymmetric with a different number of LSTM units. Indeed, the network design is data-driven, and according to Auto ML principles, the DT also performs an automatic Grid-search over the network's hyper-parameters (such as the number of LSTM units, number of epochs, batch size, and learning rate) and stores this information in the ML-Metadata and Logging module.

One challenge when dealing with such tasks with unsupervised techniques is teaching the model that the same trajectory can be, for example, normal and allowed if the user belongs to the medical staff but not if he belongs to another category without supervising the training. Figure 9 shows a subset of normal and anomaly trajectories for each class of user inside the real hospital ward. As it is possible to see in that figure, the anomaly trajectories performed by users share a large part of the path with the ones performed by normal users, especially for the doctors that are allowed to access to most of the environment. In light of this challenge, we provide the categorical variable that indicates the user's category in input to the encoder along with its trajectory to condition the learning of such differences.

The benefits of adopting the unsupervised learning paradigm to detect anomalies in the complex scenario of Healthcare can be summarized in the following:

- **No need of human supervision to label each trajectory:** Path followed by the users are only divided on the basis of the user's role in the organization. Training samples can be automatically collected by trusted users for each role.
- **Generalization:** The same approach can be directly applied in a different hospital or scenario thanks to the Digital Twin abstraction and a novel training phase.
- **Flexibility:** If something changes in the environment (e.g., layout changes for organizational needs or temporary changes), it is sufficient to ask the infrastructure to repeat the training using the most recent data of trusted users.

5.3. AutoML

The intelligent DT plays a fundamental role in detecting changes in the environment and performance monitoring that may induce the need for a new ML training phase. This AutoML features also have the duty of performing an automatic neural network design procedure that aims to identify the most promising hyper-parameters for the ML model, for example, the number of units for the encoder network (n) and the number of units for the decoder network

(k). These values are automatically identified by an intelligent node that has the duty of performing a Grid-search over these two parameters. The I-DT performs the hyper-parameters search whenever a new sufficient batch of trusted users is collected or during the evolution of the model when the new task should be learned. In our case, a new task is represented by the presence of trajectories from a new category of users and, thus, data detected with another IoT technology in the physical world. The AutoML module evaluates several combinations of encoder and decoder units, trying to minimize the Root Mean Squared Error between the input sequence and the reconstructed one. Once the best model has been identified in terms of average RMSE for trajectory reconstruction, all the parameters are stored by the DT in the Metadata and Logging component for further analysis and the next comparisons when the model requires new training. The final number of units for the encoder and decoder identified by the DT when all the trajectories of trusted users were available for each category are $n = 900$ and $k = 800$.

Once the optimal structure is identified and trained to minimize the average RMSE overall training sequences, the I-LBS identifies an RMSE threshold on the validation set to define when a sequence should be classified as an anomaly or not. In light of this, the I-LBS computes the RMSE over the Validation-set, assuming a Gaussian distribution of the error as null hypothesis $\mathcal{N}(\mu, \sigma^2)$.

Where μ is the average RMSE on the validation set reached during the training phase. Since the validation set is a subset of the training set composed of only normal (correct) behaviours from trusted users, the RMSE shows the ability of the network to reconstruct correct trajectories. On the other hand, we expect that the RMSE over a wrong and anomaly trajectory is usually higher than the one on correct trajectories. The I-LBS estimates a threshold using the validation set as the expected value of the distribution plus a multiplier of the standard deviation. More high the threshold, the more good trajectories will be recognized as normal. However, since the same path can be considered an anomaly, for example, for a patient but not for a clinician, some wrong trajectories could be reconstructed with RMSE close to the expected value of the Gaussian distribution. In light of this, the threshold selection node fits a Gaussian distribution over the validation-set RMSE. The threshold is then selected as $\tau = \mu + m * \sigma$, where the m defines a multiplier for the σ to be used as a threshold. An example of the result of the threshold selection for this use-case is presented in Figure 10 that shows the distribution of the reconstruction error for the Validation-set in terms of RMSE and how the anomalies' threshold (red bar) is selected by the AutoML module according to the process previously described using the Gaussian distribution over the errors. In this case, when a new trajectory is reconstructed with an error above the threshold, the trajectory will be considered an anomaly. The Neural network is trained for 1000 epochs using the Early-stopping technique and the Adam Optimizer. Once the model has been trained, it is stored in the ML Model Registry for

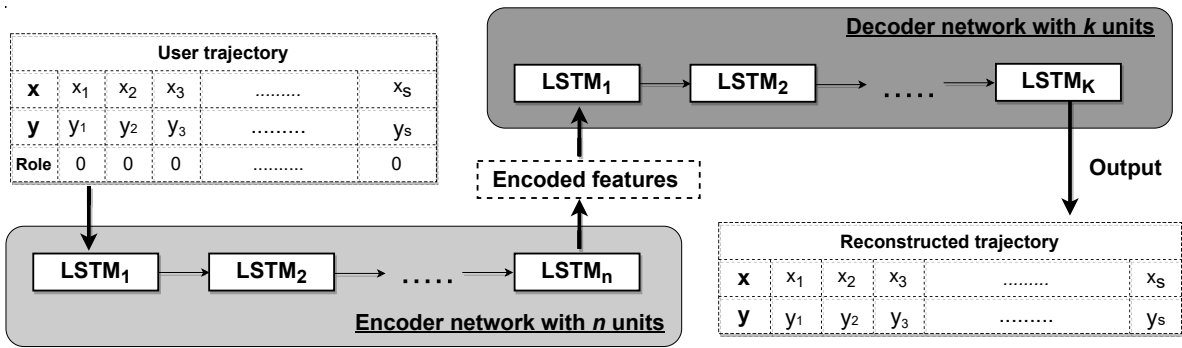


Figure 8: LSTM Encoder-Decoder architecture used to reconstruct each trajectory. The image also shows the structure we adopted for the input and the desired output.

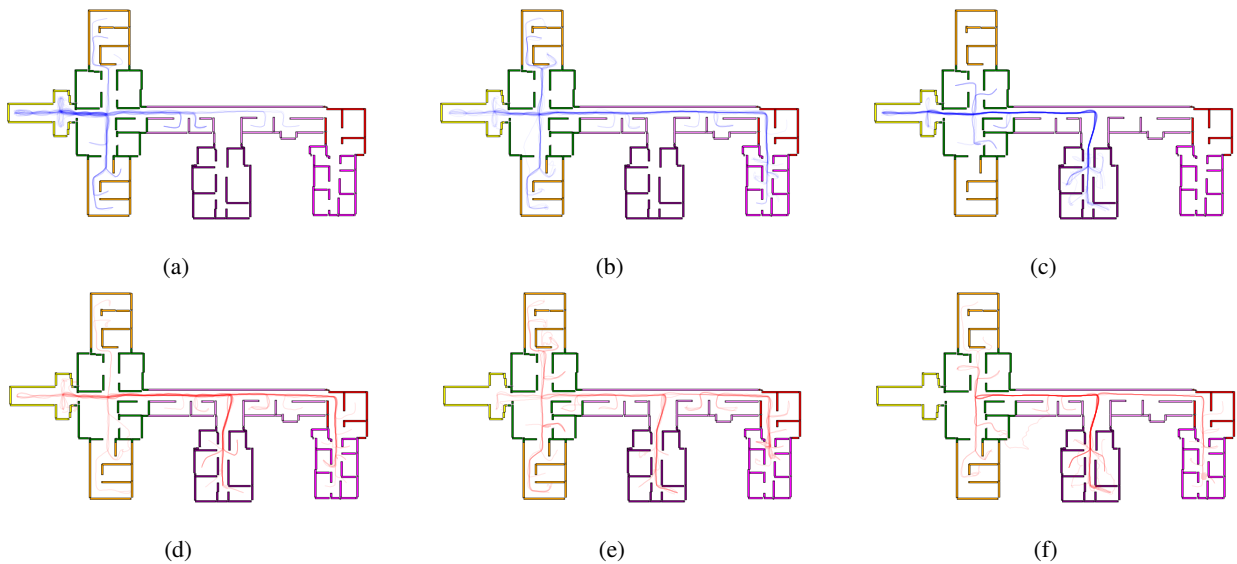


Figure 9: (a,b,c) are respectively examples of the trajectories performed by trusted patients, doctors and maintenance staff; (d,e,f) are respectively examples of the anomaly trajectories performed by patients, doctors and maintenance staff.

subsequent usage in Inference or for new training if new data become available.

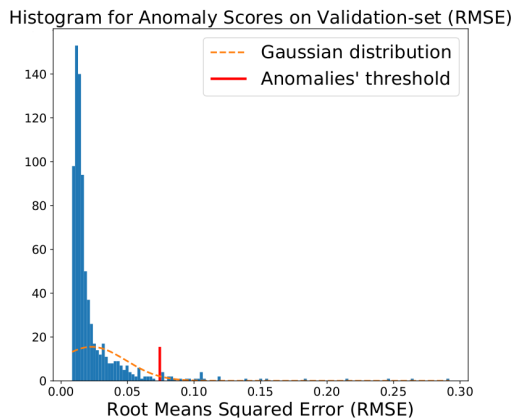


Figure 10: AutoML definition of the anomaly scores on the reconstruction error distribution for the Validation-Set.

5.4. Continual Learning over time

As a separate subcase, we also evaluated the architecture on the same anomaly detection task with a continual learning paradigm where the I-LBS has to deal with tasks and data that becomes available at different moments and thus adapts to the new scenario when previous data are not completely available anymore for new training. This subcase corresponds to the scenario where the organization may want to introduce a new role/category of users to be tracked and identified and the DT implements the MLOPs pipelines to enable the model, and thus the intelligent service, to evolve and adapt to changes in the physical world. The model is trained over time by adopting a Continual Lifelong learning paradigm. At the beginning, the DT architecture is only exposed to users belonging to the doctors' category. The trusted users' trajectories are collected for training, and then I-DT evaluates the performance on a validation set. After that, the environment is populated with maintenance staff users. This novelty requires the architecture to adapt the intelligent service with new training since the number of possible users is increased, as well as the knowledge

about map segmentation. Finally, the environment is also populated with patients.

In light of this, the model is first trained to only reconstruct normal trajectories of doctors, which represents the first task. After that, the neural network is trained to deal with a second task: reconstructing normal trajectories of the maintenance staff and, finally, the patients' trajectories. The main goal of this experiment is to avoid and reduce the so-called Catastrophic Forgetting of the model when learning new tasks. In our case, this goal becomes more challenging since all tasks are learned in an unsupervised way. The I-LBS Digital Twin implements a Replay-based method by analyzing the performances when changing the percentage of replay samples ($R\%$). The method is composed of the following steps:

1. We first trained the model using only the 100% of doctors' trajectories available using the same parameters and 1/3 of the epochs used in the previous training (Traditional training) retrieved from the Metadata logging component.
2. Starting from the previous model, the neural network is trained to learn the second task (reconstruct normal trajectories of maintenance staff), by using the 100 % available samples for this new task and a number of examples of the previous task equal to $R\%$
3. Finally, the third task (reconstructing normal trajectories of patients) is learned in the same way as the second one: 100 % available samples for the new task and some additional examples for the previous tasks to avoid the Catastrophic Forgetting of the network. Respectively, the $R\%$ of the available doctors' normal trajectories and of the available maintenance normal trajectories.

6. Experimental evaluation

In this section, we presented the main results achieved by experimenting with the DT architecture in our use case. Results can be divided into two macro-categories:

- Results related to the platform in terms of performance and reduction of physical complexity measured with a proposed physical complexity index
- Results related to the intelligent location-based service in terms of detection rate of allowed/not allowed trajectories in a normal scenario and in a continual learning one with a manually created test set that includes normal trajectories and anomaly trajectories.

6.1. Measuring Cyber-Physical Benefits

As anticipated in previous sections, one of the main advantages of the proposed approach is the possibility to simplify and manage the heterogeneity and complexity of the physical layer through the adoption of Digital Twins. In this context and with respect to the target experimental use case, it is important to try to estimate and measure the physical complexity with and without the adoption of

the proposed architecture in order to better understand the introduced benefits in terms of digitalization and decomposition of responsibilities between DTs (in charge of handling physical assets) and intelligent services (focusing only on data processing).

In this section, we aim to measure the positive impact of DTs in terms of their capabilities to uniform the existing physical fragmentation providing a new digital, interoperable, and homogenous representation for the designed upper layers and I-DTs. In order to address this task, we introduce the *Physical Complexity Index* (PCI) as a simple indicator to measure the impact of a set of criteria affecting the management of the heterogeneity and complexity characterizing a real-world deployment. With respect to our experimental use case, we identified the following criteria:

- *Required Protocols*: The number of application layer protocols adopted to talk with deployed physical assets (in our scenario MQTT and HTTP);
- *Communication Patterns*: The number of patterns (e.g., Pub/Sub, Request/Response or RESTful) required to interact with involved devices and platforms. In our use case without DTs we have Pub/Sub and RESTful at the same time, while with the introduction of DTs, only MQTT;
- *Data Formats*: The number of different data formats and serialization/deserialization techniques required to talk with the physical layer. In our scenario, without DTs we have three different JSON (JavaScript Object Notation) formats, one for each LBS platform, and with DTs we have only one uniform data format;
- *Interaction Points*: The number of different modules, services or platforms that an application should interact with to retrieve all the target data. In our use case without DTs we have three different LBS platforms deployed on independent cloud facilities, while with DTs we obtain a single interaction point.;
- *Disambiguation Points*: The number of disambiguation steps required to uniform the platform knowledge in case the same physical entity can be represented or mapped through multiple platforms at the same time. In our experimental evaluation, an entity (e.g., a doctor) may be detected through multiple LBS platforms at the same time through different physical radio interfaces of devices (e.g., WiFi, Bluetooth, or RFID). Without DTs, we have three different disambiguation points, one for each LBS platform, while with DTs, the disambiguation is managed internally, and the intelligent layer does not require any additional processing steps.

Each parameter is then associated with a specific Criteria Importance Factor (CIF) ranging from 1 (lower) to 3 (higher) used to increase the weight of specific criteria that may impact the development, deployment, and maintenance of

Criteria	CIF	Without DT	Without DT & CIF Correction	With DT	With DT & CIF Correction
Required Protocols	2	2	4	1	2
Communication Patterns	1	2	2	1	1
Data Formats	3	3	9	1	3
Interaction Points	2	3	6	1	2
Disambiguation Points	3	3	9	0	0
Results	-	13	30	4	8

Table 1

Estimation of the Physical Complexity Index for the target use case in order to evaluate through the combination of multiple parameters and the relative Criteria Importance Factor (CIF) the benefit for the intelligent layers provided by the introduction of DTs to simplify the interaction with the physical world.

the solution. The PCI is directly proportional to the exposed physical complexity that an external intelligent application is in charge of managing in order to communicate with deployed physical assets, collect data and send commands.

Relying on the defined PCI, we applied it to our experimental scenario described in Section 5 in order to compare the exposed physical complexity with and without the use of the designed and deployed DTs and considering identified CIFs according to the relevance of each factor in the target simulated environment. Table 1 and Figure 11 report the evaluation of the PCI and Depicted values properly depicting and measuring how the introduction of DTs brings an effective reduction of the physical complexity allowing an intelligent service to interact with a uniform digital interface and to focus only on data processing and the generation of intelligent insights. In particular, even if we are conducting experiments in a simulated and controlled scenario with a reduced number of protocols and data formats, the implementation and deployment of DTs significantly reduce the number of disambiguation points, limit the adoption of fragmented data formats and enable the adoption of a shared and standard protocol for cyber-physical interactions. The following sections will instead be focused on the evaluation of the proposed platform in terms of computational performance, introduced costs, and the effectiveness of the provided intelligent capabilities.

6.2. Platform Performance

Relying on the defined virtualized environment we designed and built two different DTs associated respectively to Cisco and Extreme Network LBS platforms and responsible for the management of the target locations and directly handling incoming received messages and callbacks. At the same time, a prototype of the I-DT has been developed in order to handle the disambiguation of the incoming information, provide a uniform representation of the areas of interest and augment with intelligent capabilities. The evaluated DTs have been implemented using the WLDT library⁶, a modular Java software stack based on a shared multi-thread engine able to effectively implement DT behavior and to define its mirroring procedures, data processing, and the interaction with external applications Picone et al. (2021a). The first group of DTs receives raw information from emulated LBS

⁶White Label Digital Twin (WLDT) - GitHub - <https://github.com/wldt>

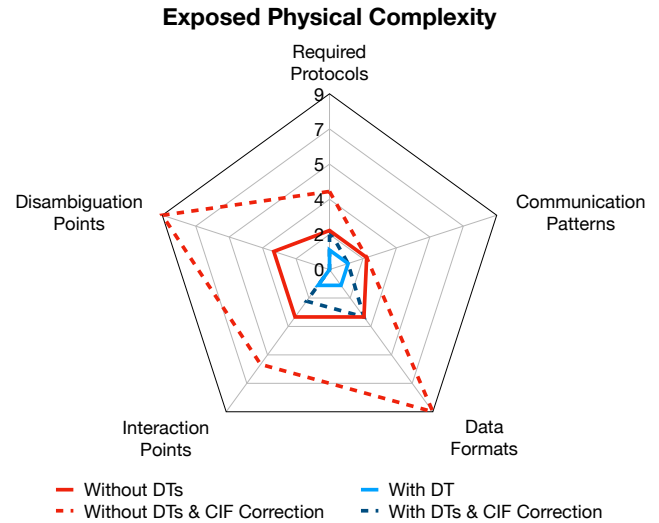


Figure 11: Graphical representation of the Physical Complexity Index for the target application scenario taking into account results with and without the correction of the Criteria Importance Factor (CIF).

platforms using RESTful HTTP APIs and data structured equivalent to the originals. It publishes enriched and uniform data to the I-DT through a Pub/Sub approach implemented through the MQTTBanks and Gupta (2014) protocol.

The intent of this evaluation step is to understand the computational cost introduced by the DTs in terms of CPU, Memory Consumption, and End-to-End (E2E) Delay from the original data generation of it available on the event communication layer. Performed tests have been executed on a Linux node equipped with 2,3 GHz Quad-Core Intel Core i7 and 32 GB of RAM through averaging on a sequence of multiple runs of 15 minutes, taking into account three different messages rates for the emulated LBS platforms of $1msg/sec$, $25msg/sec$ and $50msg/sec$. The DT has been executed as a native Java process with a memory limit of 64 MBytes.

Figure 12 (a) and (b) report an exemplifying timeline with the CPU and Memory consumption over a period of 15 minutes and with respect to the three identified message rates. Graphs show how the implemented DT is able to properly handle the incoming data load with a limited load

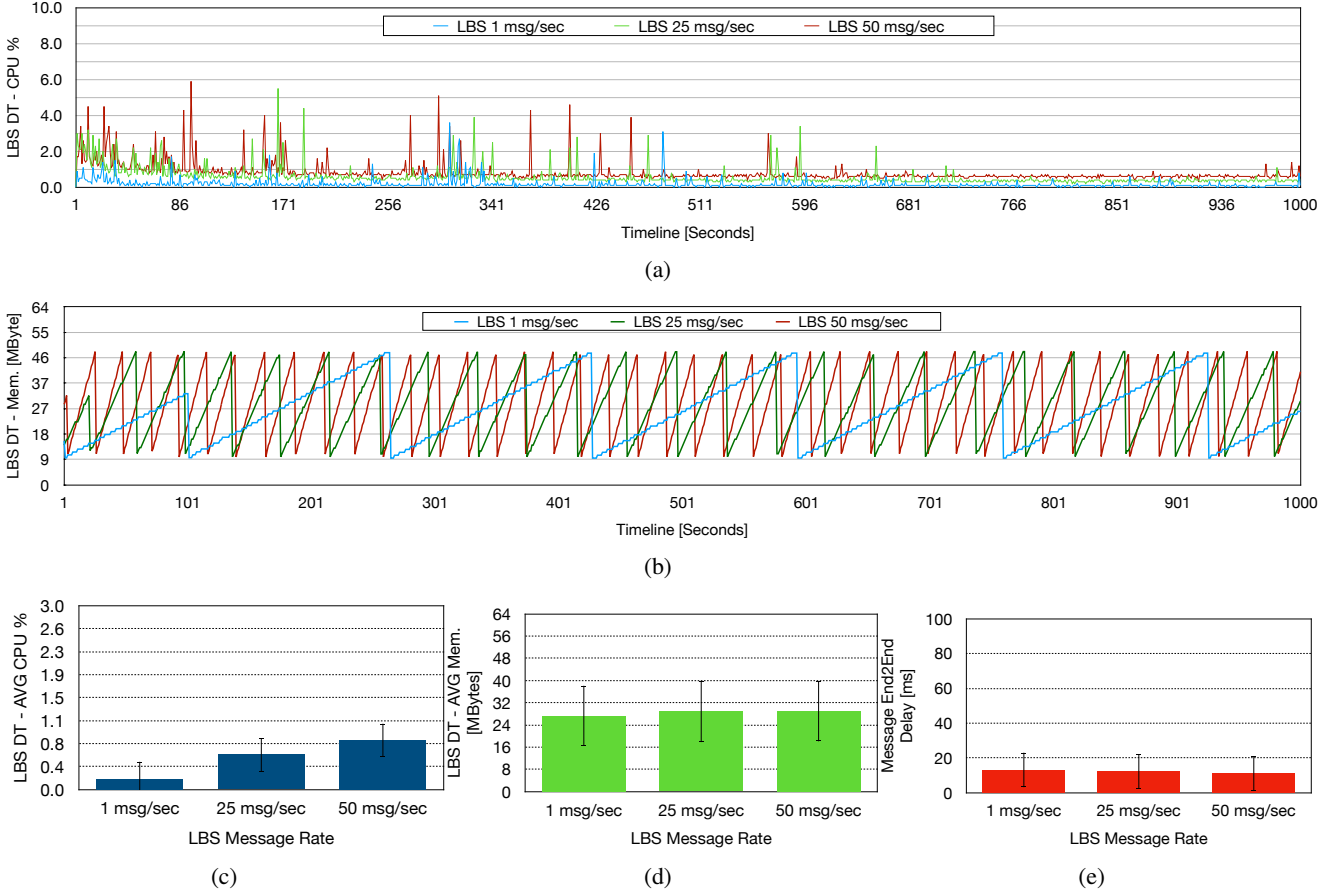


Figure 12: (a,b) Loads in terms of CPU % and Memory for the target LBS Digital Twin over an experimental timeline; (c,d) Average CPU and Memor load with respect to different message rates; and (e) The average end-to-end delay from lbs data generation to DT transformation and disambiguation.

in terms of CPU and memory usage with the characteristic trend of the Java Garbage Collector⁷.

The same metrics have been averaged on multiple independent runs and reported in the histograms of Figures 12 (c) and (d). As expected, load trends are confirmed with a limited DT resource consumption and only a slight increase in the metrics due to the higher message rate.

Graphs in Figure 12 (d) are instead focused on analyzing the introduced E2E Delay required by the DTs to process incoming LBS messages (containing raw localization information) and generate a new enriched, structured, and uniform message for the other architectural modules. Obtained results show how the introduced delay is on average, around 15ms, and the DTs are able to keep the same performance even if there is an increased message rate from the physical layer.

6.3. Anomaly detection performances with a traditional setting

In this section, we present the results achieved with the I-LBS when it is deployed in a traditional setting with

⁷Oracle Java Garbage Collector - <https://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html>

a prior knowledge of all the users' categories that should be recognized. Moreover, we provide a sensitive analysis of the m parameter (error threshold) that is evaluated by the AutoML module on a validation set. The test set, used to evaluate the generalization capability of the I-LBS with unknown trajectories, is evaluated on 50 independent runs. For every run, the test is built according to the following procedure:

- We collected 370 examples of allowed and normal paths for each class of users.
- 370 examples of forced anomaly trajectories for each class with an additional random behaviour that tries to correct these trajectories towards a similarity to the normal ones using the attractive methods provided by the Boid model.
- Each trajectory is also chosen by randomly selecting among a buffer of paths obtained with a sliding and overlapping window mechanism. In this way, we can evaluate trajectories also considering when the model knows their starting point, in the intermediate steps and in the final ones.

- These technical choices are made up for the sake of the evaluation, aiming to create anomaly and not anomaly examples that can be harder to be recognized by the model.
- We achieved the same number of examples for anomaly and allowed trajectories, and the test set is also balanced in terms of examples of patients, medical staff, and maintenance.

The best model identified by the AutoML module reached an average **RMSE=0.03439** on the validation set with an error distribution $\mathcal{N}(\mu, \sigma^2)$ with $\mu = 0.022$ and $\sigma = 0.026$. When the reconstruction error is above that value, the trajectory is classified as an anomaly.

Defining the multiplier m is not trivial because different choices can be made depending on which kind of detection error could be tolerated and which one is required to be minimized. Remembering that these parameters are estimated on a validation set built with only positive (normal) trajectory examples, one approach to study this trade-off is computing the Recall on that class as a function of the multiplier m . Figure 13 (left) shows the fraction of correct trajectories that are correctly classified as normal by varying the multiplier for the threshold on the validation set. It is evident that by selecting a multiplier equal to 3, most of the trajectories are correctly classified as normal (97.8%). However, such a high threshold would also recognize the most difficult anomaly trajectories as normal. On the other hand, a threshold equal to the standard deviation ($m = 1$) would classify an important number of correct trajectories in the distribution tail as an anomaly. For this reason, we selected, as a trade-off for the I-LBS, a multiplier $m = 2$. In this way, we want to deal with the importance of having a low number of false negatives (normal trajectories that are detected as anomalies) by keeping as much as possible the ability of the model to detect anomalies for each role in the organization.

With $\tau = \mu + 2\sigma = 0.072$, the model reached an overall accuracy equal to **91.08%** on a balanced test-set, the confusion matrix is reported in Table 2 (B). For sake of completeness, we also report the confusion matrix that would be obtained with a different choice of the multiplier m , see Table 2 (A) and (C). It is clear that the multiplier m affects, for a small part, the final accuracy on the test set but can be tuned for the desired behaviour depending on the impact that false negatives and false positives may have in the organization. Finally, Figure 13 shows the distribution of the RMSE on the test set for normal and anomaly trajectories and the numerical impact of the threshold.

6.4. Continual Learning performances

In this section, we analyze the results achieved with our proposed DT architecture on the Continual Learning side. We report the following results:

1. The I-LBS intelligence performance when users' roles become available only during the time and require an adaptive AI approach with Continual Learning. At

the beginning, the model is only exposed to doctors' trajectories. Secondly, the maintenance staff's trajectories and finally, the model should learn how to also recognize patients' trajectories.

2. The results achieved with the same AutoML module when the ML model is trained with a traditional setting where all the tasks are known at training (See Section 6.3).

For each configuration, we report an average of over 50 different runs. We considered the Continual Learning setting presented in Section 5.4 by leveraging the Replay-based method, which exploits a variable amount of samples of the previously learned tasks while learning a novel task (replay percentage $R_{\%}$). Performances are measured in terms of Precision, Recall and F-1 score for the three binary tasks of anomaly detection for each user's role in the hospital (Patients, Doctors and Maintenance staff). The accuracy is measured over the overall classification task (anomaly/not anomaly).

If trained with a replay percentage $R_{\%}$ equal to zero, Catastrophic Forgetting occurs, and only the third task is learned. Introducing a $R_{\%} = 1\%$, performances improve with a final accuracy in the binary task (normal/anomaly) equal to 62.7%. At the level of the single-user category, the one that suffers mostly a bad performance is the one related to the maintenance staff.

We then tested with $R_{\%} = 10, 30, 50\%$. Interestingly, by only replaying the 10% of examples of the first two tasks, the final binary accuracy reaches, on average, 84.5%, and the slope of improvements at the single categories level becomes less pronounced.

Finally, we tested using $R_{\%} = 100$ to compare the traditional training with a pure incremental (online) one. Recalling that the traditional model with the same parameters achieved on average a binary accuracy equal to 91.08, it is fair that the incremental one reaches, on average, an accuracy equal to 90.8%. It is clear that there is not any important difference in training incrementally over the single categories if the replay involves all the available examples.

Figure 14 shows the F1-score, Precision and Recall for the anomaly class for each task by varying the replay percentage $R_{\%}$. Recalling that the learning order was doctors, maintenance and at the end, the patients, Figure 14 shows that, in terms of F-1 score, the ability to recognize anomalies for the first learned classes (doctors and maintenance) do not degrade when learning the third one with only a Replay percentage equal to the 30%, if compared with the results achieved with a traditional learning setting. However, in terms of F-1 score, the patients' category is slightly learned better with a traditional learning paradigm.

7. Discussion & Future Works

In this research work, we propose a Digital Twin based platform that aims to cover the current lacks when developing intelligent services from Location Based Services data. We highlighted how the introduction of DTs can provide an

Actual class	Prediction outcome		Actual class	Prediction outcome		Actual class	Prediction outcome	
	Normal	Anomaly		Normal	Anomaly		Normal	Anomaly
Normal	329	41	Normal	348	22	Normal	363	7
Anomaly	30	340	Anomaly	44	326	Anomaly	60	310
τ	m	Accuracy	τ	m	Accuracy	τ	m	Accuracy
0.048	1	90.4%	0.072	2	91.08%	0.097	3	90.9%

Table 2

Sensitivity analysis of the average confusion matrices and accuracies with A) $m = 1$; B) $m = 2$; C) $m = 3$. Elements along the diagonal are the ones correctly classified.

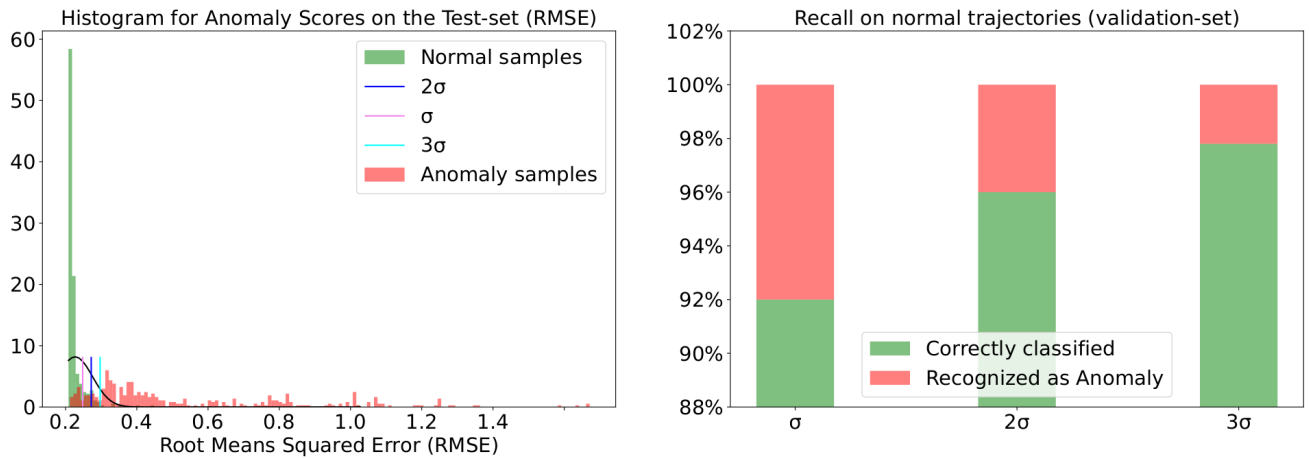


Figure 13: (Left): Histogram for anomaly scores on the Test-set highlighting normal and anomaly samples. (Right): Recall for normal trajectories on the validation set depending on the selected threshold.

effective physical abstraction to the I-LBSs by reducing the physical complexity and allowing an intelligent service to interact with a uniform digital interface. Machine Learning and Deep Learning can benefit from that by focusing only on data processing and the generation of intelligent insights by leveraging an execution graph of ML operations. Moreover, the platform can concretely support the development of Continual Learning tasks in this domain, providing the necessary flexibility and generalization. Finally, we proposed a use-case for this platform in a complex healthcare organization and management scenario where the I-LBS classifies the

high-resolution trajectories of users inside a real-existing hospital scenario, depending on their role in the organization. The use case is implemented as a Deep Learning unsupervised reconstruction task thanks to the platform, and it also enables a comparison between a traditional and a CL approach for this task, demonstrating that it can be a solution, especially when flexibility and adaptability of the service to dynamic or unseen scenarios is needed. Future works are related to improving this platform in other domains by leveraging other advances in Deep Learning and plan to support the other CL methodologies and techniques for supervised

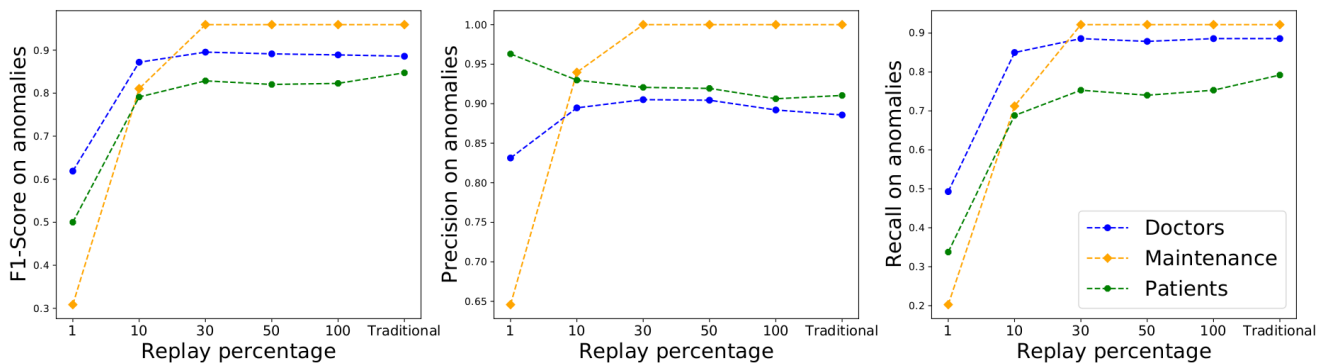


Figure 14: F-measure, Precision, and Recall for each task when using a Replay method for the Continual Learning scenario. Tasks are learned in the order: Doctors, maintenance, and finally, patients. Results are reported as an average of different runs with respect to the replay percentage used.

learning. Moreover, we will evaluate the applicability and extension of the proposed approach to additional use cases such as outdoor mobility for Smart City and anomaly detection in industrial scenarios. Nevertheless, we will investigate the possibility of generalizing the proposed approach (or its main component) to extend its applicability to multiple application domains by maximizing infrastructure, modules, and code reusability.

CRedit authorship contribution statement

Gianfranco Lombardo: Conceptualization of this study, Methodology, Software, Writing - Original draft preparation. **Marco Picone:** Conceptualization of this study, Methodology, Software, Writing - Original draft preparation. **Marco Mamei:** Validation, Methodology, Formal analysis, Writing - Original draft preparation, Supervision. **Monica Mordonini:** Writing - Review and Editing, Resources, Supervision. **Agostino Poggi:** Software, Formal analysis, Writing - Review and Editing, Supervision.

References

- Afyouni, I., Ray, C., Claramunt, C., 2017. Representation: indoor spaces. International Encyclopedia of Geography: People, the Earth, Environment and Technology: People, the Earth, Environment and Technology , 1–12.
- Agha, G., 1986. Actors: a model of concurrent computation in distributed systems. MIT press.
- Angiani, G., Fornacciari, P., Lombardo, G., Poggi, A., Tomaiuolo, M., 2018. Actors based agent modelling and simulation, in: International Conference on Practical Applications of Agents and Multi-Agent Systems, Springer. pp. 443–455.
- Autiosalo, J., Siegel, J., Tammi, K., 2021. Twinbase: Open-source server software for the digital twin web. IEEE Access 9, 140779–140798. doi:10.1109/ACCESS.2021.3119487.
- Banks, A., Gupta, R., 2014. MQTT Version 3.1.1. URL: <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html>.
- Barricelli, B.R., Casiraghi, E., Fogli, D., 2019. A survey on digital twin: Definitions, characteristics, applications, and design implications. IEEE Access 7, 167653–167671. doi:10.1109/ACCESS.2019.2953499.
- Bellavista, P., Giannelli, C., Mamei, M., Mendula, M., Picone, M., 2021. Application-driven network-aware digital twin management in industrial edge environments. IEEE Transactions on Industrial Informatics .
- Bergenti, F., Poggi, A., Tomaiuolo, M., 2014. An actor based software framework for scalable applications, in: International Conference on Internet and Distributed Computing Systems, Springer. pp. 26–35.
- Buzzega, P., Boschini, M., Porrello, A., Abati, D., Calderara, S., 2020. Dark experience for general continual learning: a strong, simple baseline. Advances in neural information processing systems 33, 15920–15930.
- Eramo, R., Bordeleau, F., Combemale, B., van den Brand, M., Wimmer, M., Wortmann, A., 2021. Conceptualizing digital twins. IEEE Software , 0–0doi:10.1109/MS.2021.3130755.
- Fujii, T.Y., Hayashi, V.T., Arakaki, R., Ruggiero, W.V., Bulla Jr, R., Hayashi, F.H., Khalil, K.A., 2021. A digital twin architecture model applied with mlps techniques to improve short-term energy consumption prediction. Machines 10, 23.
- Gupta, R., Rao, U.P., 2017. An exploration to location based service and its privacy preserving techniques: a survey. Wireless Personal Communications 96, 1973–2007.
- Hashash, O., Chaccour, C., Saad, W., 2022. Edge continual learning for dynamic digital twins over wireless networks. arXiv preprint arXiv:2204.04795 .
- Huang, H., Gartner, G., Krisp, J.M., Raubal, M., Van de Weghe, N., 2018. Location based services: ongoing evolution and research agenda. Journal of Location Based Services 12, 63–93.
- Jiang, H., Li, J., Zhao, P., Zeng, F., Xiao, Z., Iyengar, A., 2021. Location privacy-preserving mechanisms in location-based services: A comprehensive survey. ACM Computing Surveys (CSUR) 54, 1–36.
- Kemker, R., McClure, M., Abitino, A., Hayes, T., Kanan, C., 2018. Measuring catastrophic forgetting in neural networks, in: Proceedings of the AAAI conference on artificial intelligence.
- Kolltveit, A.B., Li, J., 2022. Operationalizing machine learning models—a systematic literature review, in: 2022 IEEE/ACM 1st International Workshop on Software Engineering for Responsible Artificial Intelligence (SE4RAI), IEEE. pp. 1–8.
- Koulamas, C., Kalogeras, A., 2018. Cyber-physical systems and digital twins in the industrial internet of things [cyber-physical systems]. Computer 51, 95–98. doi:10.1109/MC.2018.2876181.
- Kreuzberger, D., Kühl, N., Hirschl, S., 2022. Machine learning operations (mlps): Overview, definition, and architecture. arXiv preprint arXiv:2205.02302 .
- Kumara, I., Arts, R., Di Nucci, D., Heuvel, W.J.V.D., Tamburri, D.A., 2022. Requirements and reference architecture for mlps: Insights from industry .
- Lee, J., Joo, D., Hong, H.G., Kim, J., 2020. Residual continual learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 4553–4560.
- Lee, S.W., Kim, J.H., Jun, J., Ha, J.W., Zhang, B.T., 2017. Overcoming catastrophic forgetting by incremental moment matching. Advances in neural information processing systems 30.
- Leroux, S., Simoons, P., Lootus, M., Thakore, K., Sharma, A., 2022. Tinymlps: Operational challenges for widespread edge ai adoption, in: 2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), IEEE. pp. 1003–1010.
- Lombardo, G., Pellegrino, M., Poggi, A., 2022a. Unsupervised continual learning from synthetic data generated with agent-based modeling and simulation: A preliminary experimentation .
- Lombardo, G., Poggi, A., Tomaiuolo, M., 2022b. Continual representation learning for node classification in power-law graphs. Future Generation Computer Systems 128, 420–428.
- Maltoni, D., Lomonaco, V., 2019. Continuous learning in single-incremental-task scenarios. Neural Networks 116, 56–73.
- Matsui, B.M., Goya, D.H., 2022. Mlps: five steps to guide its effective implementation, in: Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI, pp. 33–34.
- Minerva, R., Crespi, N., 2021. Digital twins: Properties, software frameworks, and application scenarios. IT Professional 23, 51–55. doi:10.1109/MITP.2020.2982896.
- Minerva, R., Lee, G.M., Crespi, N., 2020. Digital twin in the iot context: A survey on technical features, scenarios, and architectural models. Proceedings of the IEEE , 1–40.
- Parisi, G.L., Kemker, R., Part, J.L., Kanan, C., Wermter, S., 2019. Continual lifelong learning with neural networks: A review. Neural Networks 113, 54–71.
- Picone, M., Mamei, M., Zambonelli, F., 2021a. Wldt: A general purpose library to build iot digital twins. SoftwareX 13, 100661. URL: <https://www.sciencedirect.com/science/article/pii/S2352711021000066>, doi:<https://doi.org/10.1016/j.softx.2021.100661>.
- Picone, M., Mariani, S., Mamei, M., Zambonelli, F., 2021b. Wip: Preliminary evaluation of digital twins on mec software architecture, in: IEEE 22nd International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM). In press.
- Radanliev, P., De Roure, D., Nicolescu, R., Huth, M., Santos, O., 2022. Digital twins: artificial intelligence and the iot cyber-physical systems in industry 4.0. International Journal of Intelligent Robotics and Applications 6. doi:10.1007/s41315-021-00180-5.
- Ricci, A., Croatti, A., Mariani, S., Montagna, S., Picone, M., 2021. Web of digital twins. ACM Transaction on Internet Technology. URL: <https://doi.org/10.1145/3507909>, doi:10.1145/3507909.

- Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., Wayne, G., 2019. Experience replay for continual learning. *Advances in Neural Information Processing Systems* 32.
- Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y.W., Pascanu, R., Hadsell, R., 2018. Progress & compress: A scalable framework for continual learning, in: *International Conference on Machine Learning*, PMLR. pp. 4528–4537.
- Shin, H., Lee, J.K., Kim, J., Kim, J., 2017. Continual learning with deep generative replay. *Advances in neural information processing systems* 30.
- Tadros, T., Krishnan, G.P., Ramyaa, R., Bazhenov, M., 2022. Sleep-like unsupervised replay reduces catastrophic forgetting in artificial neural networks. *Nature Communications* 13, 7742.
- Tao, F., Zhang, H., Liu, A., Nee, A.Y.C., 2019. Digital twin in industry: State-of-the-art. *IEEE Transactions on Industrial Informatics* 15, 2405–2415. doi:10.1109/TII.2018.2873186.
- Tao, F., Zhang, M., Nee, A., 2019. Chapter 1 - background and concept of digital twin, in: Tao, F., Zhang, M., Nee, A. (Eds.), *Digital Twin Driven Smart Manufacturing*. Academic Press, pp. 3 – 28. doi:https://doi.org/10.1016/B978-0-12-817630-6.00001-1.
- Wang, L., Zhang, X., Su, H., Zhu, J., 2023. A comprehensive survey of continual learning: Theory, method and application. *arXiv preprint arXiv:2302.00487*.
- Yu, Y., Si, X., Hu, C., Zhang, J., 2019. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation* 31, 1235–1270.
- Zenke, F., Poole, B., Ganguli, S., 2017. Continual learning through synaptic intelligence, in: *International Conference on Machine Learning*, PMLR. pp. 3987–3995.
- Zhao, Z., Shen, L., Yang, C., Wu, W., Zhang, M., Huang, G.Q., 2021. Iot and digital twin enabled smart tracking for safety management. *Computers & Operations Research* 128, 105183.