Future Generation Computer Systems 185 (2026) 108674

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Digital Twin Aggregates for adaptive MLOps retraining policies in healthcare☆

Davide Domini [a,*], Leonardo Micelli [b], Samuele Burattini [a], Sara Montagna [b]

[a] Alma Mater Studiorum - University of Bologna, Via dell'Università, 50, 47522 Cesena FC, Italy
[b] University of Urbino, Via Aurelio Saffi, 2, 61029 Urbino PU, Italy

ARTICLE INFO

Keywords:
MLOps
Digital Twin Aggregate
Healthcare

ABSTRACT

Digital Twins (DTs) are increasingly adopted as a technical solution for the digital representation of complex physical entities through models that process near-real-time data streams and provide feedback on the state and behavior of the Physical Twin (PT). When DT models are trained using Machine Learning (ML) techniques, their performance may degrade over time as the DT acquires new operational data that may differ from the data observed during initial training. In this paper, following Machine Learning Operations (MLOps) principles, we investigate how Digital Twin Aggregates (DTAs) can be integrated into DT-based systems to enable continuous monitoring of model performance and to support adaptive retraining strategies for the continuous delivery and maintenance of ML models ensuring that the DT remains a reliable representation of the PT throughout its lifecycle. We evaluate the approach in a healthcare case study involving DTs for diabetic patients and compare adaptive with periodic retraining showing that performance-based retraining maintains stable accuracy while reducing model updates. These results suggest that DTA-level monitoring enables more efficient adaptive MLOps lifecycles by triggering retraining in response to actual performance degradation rather than fixed schedules.

1. Introduction

In many clinical contexts, data are inherently dynamic, evolving both along the longitudinal trajectory of individual patients and through the continuous inclusion of newly observed subjects. This dual temporal dimension introduces substantial complexity when training purely data-driven Machine Learning (ML) models with the goal to identifying latent temporal patterns, generating predictions and supporting decision-making. Accordingly, the integration of continuously evolving data, to update predictions and refine learned representations as new evidence becomes available, remains an open challenge within ML application in healthcare, requiring ad-hoc solutions. Retraining pipelines, in particular, must balance between adaptability to new data – potentially improving model performance – and stability, in order to avoid, for instance, catastrophic forgetting and unintended regressions. Scalability and resource management further complicate this scenario, as model retraining is often computationally expensive and time-consuming. Moreover, ensuring effective monitoring and observability of models in production is still a critical issue, as it is necessary to periodically assess model performance and determine when retraining is required. In this context, the need for adaptive, continuously updated models has driven the adoption of paradigms that go beyond static ML pipelines, enabling real-time synchronization between data, models, and physical systems.

Machine Learning Operations (MLOps) and Digital Twins (DTs) have emerged as reference approaches: the former provides structured methodologies for managing the ML lifecycle in production (e.g., monitoring, retraining, and scalability) [1], while the latter enables continuous interaction between models and physical systems through real-time data integration and feedback loops [2].

In particular, DTs, initially introduced as a generic virtual representation of a physical entity continuously updated with real-time data [3], are now evolving towards rich software counterparts that provide smart services [2]. Such services may range from simple tracking of the actual state of the physical entity to smarter forms of monitoring, such as detecting anomalies and predicting future behavior of the modeled asset. Moreover, on top of DTs, Digital Twin Aggregates (DTAs) have been defined as a higher-level system representation of a set of homogeneous DT instances that allows inspecting historical data of

☆ This article is part of a Special issue entitled: 'MLOps' published in Future Generation Computer Systems.

* Corresponding author.
E-mail addresses: davide.domini@unibo.it (D. Domini), leonardo.micelli@uniurb.it (L. Micelli), samuele.burattini@unibo.it (S. Burattini), sara.montagna@uniurb.it (S. Montagna).

https://doi.org/10.1016/j.future.2026.108674
Received 30 April 2026; Received in revised form 10 June 2026; Accepted 15 June 2026
Available online 27 June 2026

all individual twins and population-level analytics, while maintaining explicit traceability to the underlying individuals. The DT concept has been defined and applied across various domains, including manufacturing, aerospace [4] and smart cities [5]. More recently, it has been adopted in healthcare [6,7], where the term Human Digital Twin (HDT) is gaining increasing relevance referring to a computational representation of an individual that integrates heterogeneous data such as vital signs, clinical and -omics results, medical history, physical examination findings, and behavioral and cognitive assessments. Accordingly, HDTs are emerging as a key concept towards realizing healthcare systems which support continuous monitoring and personalized care by collecting, standardizing and integrating heterogeneous patient data over time [8,9].

In this paper, we explore the role of the DT and DTA approaches as viable abstractions to build effective MLOps pipelines that account for and continuously adapt to evolving data landscapes. This novel DT-based framework is designed to support the development, deployment, monitoring, and continuous updating of ML models. Namely, we design a two-level MLOps pipeline. At the bottom level, individual DTs collect data, (pre)process it, and perform inferences that can immediately provide insights to their users. Through the continuous feedback loop implemented by each DT, it is possible to collect performance indicators of the ML model, validating the predicted behavior through the acquisition of new data over time. At the upper level, the DTA acts as an abstraction layer where new data – acquired through DTs – are continuously incorporated and stored. The DTA can monitor the performance of each individual DT and define rules for triggering automatic re-training and deployment of the new models on the available DT instances. With this work, we demonstrate how this framework intuitively supports real-time validation of ML models and evaluate whether adaptive policies can improve the overall performance of the MLOps pipeline, comparing different Key Performance Indicators (KPIs) of the training process.

The proposed architecture is evaluated using a real-world scenario and benchmark in the context of Type 1 Diabetes Mellitus patients [10]. In this setting, vital signs – such as glycemic levels – are measured daily over several weeks, alongside static patient-specific information. New patients are continuously incorporated throughout the entire study period (four years), as additional subjects become available. Results show that the performance-based retraining policy maintains stable predictive accuracy over time while substantially reducing the number of retraining cycles compared to time-based and population-based baselines. In particular, by triggering model updates only in response to actual performance degradation, the DTA enables a more efficient adaptive MLOps lifecycle, reducing unnecessary retraining and deployment operations without compromising model quality.

The paper is organized as follows. Section 2 introduces the relevant background on MLOps and DTs, with a deep dive on how these concepts have been adopted in the healthcare domain that we use as a motivation and validation of the proposed approach. Section 3 describes the architecture of the DTA implementing the monitoring, adaptive retraining policies and continuous delivery of ML models to individual DT. Section 4 presents the medical case study used to validate this work and the experimental results comparing different retraining policies. Section 5 position the contribution within the context of related works. Section 6 discusses current limitations of the proposed approach, challenges for its application in real-world settings and outlines a research roadmap for future works in this area. Finally, Section 7 summarizes the contribution and discusses possible future works in this area.

2. Background

Building on the limitations of static, offline ML pipelines, this section outlines the key concepts of MLOps, DTs and DTAs and discuss relevant contributions that tackle the challenge of continuously updating and retraining models embedded within DTs. Given that we use healthcare scenarios as both motivation and validation for our proposal, we also report on applications of these technologies in this domain to highlight healthcare-specific challenges that make the adoption of DT-based MLOps particularly well-suited to healthcare scenarios.

2.1. Limitations of traditional ML pipelines in healthcare

ML models have been widely adopted in many clinical contexts, for example as core components of clinical decision-support systems to support diagnosis, prognosis, and treatment planning based on historical patient data [11,12]. Moving ML from research to clinical practice introduces significant challenges, particularly in real-world clinical environments characterized by continuous changes. The most discussed issue in literature is associated with the presence of *data drift*, *i.e.*, systematic shifts over time in the statistical properties of the input data and/or target variables relative to those observed during model development, which can lead to performance degradation. These shifts may arise from changes in clinical practice, data acquisition processes, population characteristics, or measurement protocols [13,14]. However, additional challenges must be considered. Clinical data are inherently longitudinal rather than static snapshots: new measurements or follow-up observations are continuously acquired, introducing temporal dependencies and evolving data distributions that challenge standard modeling assumptions [15]. In parallel, the reference population is not fixed but dynamically evolves over time, as new patients are admitted or recruited and others exit the system, potentially altering the underlying population distribution.

Accordingly, traditional offline and static development processes, while effective for early-stage research, are often inadequate in clinical practice: models trained and validated offline may fail to generalize, maintain their efficacy over time and adapt to the above-mentioned changes and variability [16,17]. To address these limitations, there is a growing need for reliable online ML solutions capable of continuous monitoring and retraining to adapt and reflect population evolution. These requirements motivate the adoption of MLOps in healthcare-related scenarios and at the intersection with DT and DTA approaches which, by design, support the continuous integration, updating, and validation of longitudinal data streams and population dynamics, as discussed in the following.

2.2. Machine Learning Operations

MLOps has emerged as a paradigm to extend DevOps practices to the engineering of clinical ML systems [18,19]. MLOps includes a set of practices and tools aimed at automating the deployment, monitoring, and maintenance of models in production. By adopting key DevOps principles, such as continuous integration and continuous delivery, plus newly devised concepts such as continuous training and monitoring, MLOps enables an iterative evolution of ML models, thus ensuring higher accountability when using the model in a production environment. MLOps facilitates the management of the entire ML lifecycle, from data acquisition and preprocessing to deployment, monitoring, and periodic updating within real-world systems. In particular, it supports model validation and performance monitoring over time, which are essential to guarantee reliability. Fundamental properties of MLOps workflows further include reproducibility of the training process and artifact versioning, to allow rolling out new versions of the models or backtracking to previous ones with ease.

MLOps is a domain agnostic method to handle the complexity of ML workflows. For what concerns healthcare – as suggested in [1,14,20] – although ML tools are becoming an integral part of many healthcare or medical software systems, the processes that support their deployment are still emerging, making healthcare scenarios ideal candidates for adopting MLOps tools. According to the literature, model performance and drift monitoring are central topics in healthcare MLOps research,

likely due to their quantifiability and the relative ease of operationalization. In particular, data drift monitoring is often the first aspect to be addressed and is the most extensively covered in the literature, as it provides a measurable and actionable signal for maintaining model reliability over time. Most of the existing work is indeed focused on automated retraining systems that periodically collect model performance metrics and trigger retraining based on predefined performance thresholds or according to fixed-interval [13].

However, healthcare MLOps is still in its infancy [1] and several aspects need to be further discussed. In particular, MLOps workflows do not focus on the longitudinal and temporally correlated nature of data [15], which are frequently reduced to static feature representations, leading to a loss of temporal structure and clinically relevant dynamics. Existing approaches, indeed, tend to focus primarily on data drift as the main source of performance degradation, often overlooking the fact that clinical data are inherently longitudinal and evolve over time [13]. Moreover, the evolution of the reference population – due to continuous patient admission or discharge – is typically not explicitly modeled, but instead managed in a reactive manner through periodic retraining. This can result in suboptimal adaptability and limited transparency in highly dynamic domains such as healthcare.

2.3. Digital Twins and Digital Twin Aggregates

The concept of DT [3] – namely, virtual representation of a physical system continuously updated with real-time data – has been defined and applied across various domains, including manufacturing, aerospace and Industry 5.0 [21]. The DT model allows digitalizing physical assets while keeping information synchronized with the real-world counterpart, thus enabling the creation of new digital services built upon collected information.

DTAs [3] have been introduced in the early days of DT-related literature. The idea stems from the roots of DTs in product lifecycle management, in which three concepts were originally defined: *(i)* the Digital Twin Prototype (DTP) follows the design phase of a product, *(ii)* the Digital Twin Instance (DTI) follows the manufacturing and delivery processes of each individual product, and *(iii)* and the Digital Twin Aggregate (DTA) allows for inspection of trends across all the products that have been made through the aggregation of all DTIs [3].

In the more modern interpretation of DTs as a digital representation of any kind of asset [2], the concept of DTA can be revisited to include population-level analysis on assets that are different from merely products. For instance, in the context of healthcare systems, DTAs are seen as a powerful abstraction to condense population-level information into highly available data [22], to complement or substitute resource-intensive laboratory experiments or investigations with *in silico* simulations [23].

From an architectural point of view, approaches based on DTAs involve a layered system, where each physical counterpart is closely mirrored by an individual and independent DT. Each DT instance can be aggregated with the others to form DTAs, where information about individual DT is gathered and processed to obtain general and broader insight on the status of the aggregate as a whole [24].

2.4. MLOps and Digital Twins

As discussed in Section 2.2, MLOps pipelines typically struggle with being able to effectively monitor the model performance as it is used in the deployment environment. Although effective in many settings, mainstream MLOps frameworks leaves two recurrent needs of many cyber–physical and IoT-driven scenarios unaddressed: a structural connection between the ML pipeline and the physical context that produces its data, and a first-class abstraction for the *populations* of deployed models that arise when a common model is shared across many physical entities [18]. DTs offer a promising solution to these limitations. By collecting real-time data and performing inferences on them, DTs enable more responsive and context-aware model monitoring. Integrating digital replicas of physical entities allows for personalized insights, continuous tracking of evolving conditions, and adaptive learning processes. This can help better manage drift, improve model accuracy, and ensure more effective predictions over time, as well as providing new data to update the model. In particular, DTs offer first-class abstractions that meet MLOps requirements. The *physical interface* and *shadowing function* [25] of a DT implement the ingestion and standardization layer that MLOps stacks typically have to assemble manually: raw sensor data are continuously collected, semantically aligned with the domain model, and exposed through a public *digital interface*. ML models embedded as augmentation functions [26] inherit access to the DT state, history, and interfaces, so their deployment becomes the act of plugging a model into a context-rich runtime that already exposes the inputs and triggers it needs. Moreover, a DT is a faithful representation of its physical counterpart: discrepancies between predictions and observed reality – the signal that MLOps monitoring is meant to capture – are already inside the DT and have recently been formalized as forms of *semantic drift* specific to DT systems [27].

In this paper, we introduce an architecture and a modeling framework that exploits the concept of DTA to support MLOps pipelines. In particular in our proposed reference framework, we will elevate the DTA to be a first-class element of the MLOps workflow, not only for simplifying the training of models and the access to data as proposed in [28], but also to directly encapsulate the retraining policies at the DTA level. The metamodel, architecture and the evaluation on a benchmark case study are presented in the following sections.

3. A Digital Twin aggregate for adaptive MLOps

A Digital Twin Aggregate (DTA) is a higher-order entity that operates on top of a population of homogeneous DT instances, providing unified access to their collective state and allowing population-level operations beyond the scope of individual twins [3,24]. While each DT is concerned with faithfully mirroring its own physical counterpart, the DTA observes trends across the entire population, accumulates data contributed by all twins, and coordinates actions that require a system-wide view. In the context of ML-powered DTs, this population-level point of view is necessary to sustain model quality over time. An individual DT operates on the data stream of a single physical entity, while the DTA aggregates data and performance metrics from the entire DT population.

Given these responsibilities typically associated with the DTA concept, we leverage this abstraction to encapsulate MLOps practices. Namely, continuous monitoring of deployed model performance, evidence-based retraining decisions, and continuous delivery of updated models back to individual DTs, representing a natural architectural blueprint for adaptive MLOps in DT-based systems. The following subsections develop this argument concretely. Section 3.1 describes the abstract architecture and information flows that make MLOps possible. Section 3.2 details how the DTA monitors model performance across the population. Section 3.4 formalizes the concept of a retraining policy in our framework. Section 3.5 synthesizes these components into a coherent adaptive MLOps lifecycle.

3.1. Abstract architecture

Fig. 1 illustrates the abstract architecture of a DTA-based system oriented towards adaptive MLOps. At the lower level, a population of individual DTs independently interacts with their physical counterparts, collecting raw data through their physical interfaces and exposing processed, standardized state through their digital interfaces. The DTA sits above this population as an orchestration layer, interacting with individual DTs through their digital interfaces.

Three information flows define the MLOps-relevant interactions between the DTA and individual DTs:

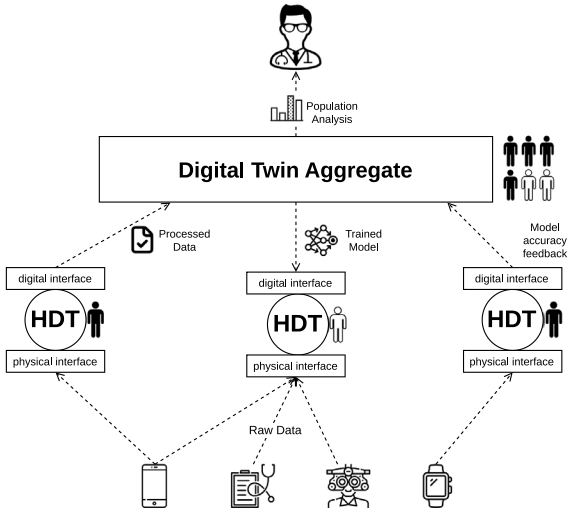

Fig. 1. Conceptual overview of the Human Digital Twin Aggregate. At the lower level, HDTs (circles) model individual patients through the collection and integration of multimodal data from heterogeneous sources. HDTs share processed data with the DTA that supports population analysis for stakeholders (top) and automatic updates of HDT models.

- *Data Aggregation (DT → DTA):* Each DT continuously contributes its processed state to the DTA. Aggregated across the population, this constitutes the dataset available for model training and retraining. The DTA is therefore the only entity in the system with access to a dataset representative of the full population — a prerequisite for training models that generalize beyond individual patients.
- *Performance Feedback (DT → DTA):* After running predictions with their local model, individual DTs can report performance of the inference model back to the DTA. The DTA aggregates the performance of each individual DT into a population-level performance estimate, which drives the monitoring and retraining logic described in the following subsections.
- *Model Delivery (DTA → DT):* When the DTA determines that retraining is warranted through a retraining policy, it trains a new ML model using the aggregated population data and can deliver it to each individual DT. Individual DTs replace their local model with the newly received one, ensuring the entire population operates with a consistent, up-to-date model at all times.

These three flows, taken together, close the MLOps loop entirely within the DTA-based system, without requiring external orchestration infrastructure.

3.2. Digital Twin state monitoring

ML for population-level analysis could benefit from the availability of a dataset that is representative of the entire DT population, not of any single physical counterpart. In the proposed system, this dataset emerges naturally from the DTA's role as an aggregation layer: each DT continuously shadows its physical counterpart, periodically reporting its current state to the DTA through its digital interface. The DTA accumulates these reports over time, constructing a growing data set that reflects the collective experience of the population.

Definition 1 (*Aggregate Data Set*)**.** Let $s^i_{t^*}$ denote the state reported by DT i at time t^*, encoding the processed and standardized representation of its physical counterpart at that moment. The data set available to the DTA at time t is then defined as: $D_t = \{s^i_{t^*} | i \in [n], t^* \leq t\}$ where n is the number of active DTs in the population.

D_t grows monotonically as new states are reported and as new DTs join the population, providing a longitudinal, increasingly rich and representative training resource throughout the system lifecycle. This accumulation mechanism is significant in the MLOps context for two reasons. First, D_t is the only dataset in the system that reflects the full population distribution — individual DTs only observe a single physical entity, which makes their local data insufficient for training models that generalize between patients. Second, D_t evolves over time as the population grows and as the conditions of individual patients change, meaning that models trained at an earlier time may no longer be representative of the current population. This temporal evolution of D_t is precisely what motivates the need for continuous monitoring and adaptive retraining, as discussed in the following subsections.

3.3. Model performance monitoring

A core requirement of MLOps is the continuous monitoring of deployed models to detect degradation before it compromises system reliability. In the DT context, this degradation can severely impact the DT's ability to accurately represent the physical asset, in our context, the human patient. For instance, *Concept drift* occurs when the statistical relationship between input features and the target variable changes over time [29]: a glucose prediction model trained on early-stage diabetic patients may become unreliable as the patient population grows to include subjects with more advanced disease progression, causing the model's assumptions to no longer hold. *Covariate shift* arises when the distribution of data used for training may differ from the actual data against which the model is used. This may happen, for instance, in cases where the analyzed population increases, revealing a different distribution from the training set, without retraining the model. Hence, in an adaptive MLOps context where population dynamically changes over time, continuous model monitoring and appropriate model retraining are key to ensuring optimal performance and trustworthiness of the system.

We consider the DTA to be the appropriate layer to implement this monitoring for two reasons. First, individual DT performance metrics are noisy when considered in isolation—a single DT could produce a run of incorrect predictions due to patient-specific anomalies rather than true model drift. Aggregating performance metrics across the population produces a more robust and statistically reliable estimate of model health. Second, individual DTs lack the system-wide view required to distinguish population-level drift from local variability.

In practice, at each timestep t, a DT i computes a set of performance metrics $\phi^i = (\phi^i_1, \ldots, \phi^i_k)$ (e.g., accuracy, precision and recall in a classification setting) plus, where applicable, locally-computed binary decision flags. Then, during its execution cycle, the DT i sends ϕ^i to the DTA. In this way, at a time step t, the DTA can compute an aggregate performance metric Φ_t for the entire DT population, for example, by averaging individual DT's metrics:

$$\Phi_t = \frac{1}{n} \sum_{1}^{n} \phi^i_t$$

where n is the size of the DT population. This aggregate monitoring state Φ_t is computed once a sufficiently representative set of individual performance metrics has been collected. Over time, the DTA accumulates the aggregate monitoring states into a set that represents the model's performance history, which serves as one of the primary inputs for the Retraining Policy.

Definition 2 (*Aggregate Performance History*)**.** Let Φ_t be the aggregate performance metric computed by the DTA at time t. We define the aggregate performance history H_t as the ordered sequence of aggregate performance metrics up to time t:

$$H_t = (\Phi_0, \ldots, \Phi_t)$$

Building on Definitions 1 and 2, we give a formalism for a *retraining policy* in the following subsection.

3.4. Retraining policies

The *Retraining Policy* is the core element in the decision-making context of a DTA for MLOps, as it is the primary way to ensure model adherence to the population. We can define a retraining policy as follows.

Definition 3 (*Retraining Policy*)**.** Let D_t be the aggregate data set (Definition 1) at time t and let H_t be the aggregate performance history (Definition 2) at time t. We define a retraining policy π as a predicate: $\pi : H_t \times D_t \to \{\textbf{retrain}, wait\}$

When the policy π evaluates to **retrain**, the DTA executes a *retraining cycle*: it trains the new model M with the data set D_t (or a subset of it) and distributes the new model to all the DTs in the population. There are different ways to define a retraining policy, from more naive policies based on elapsed time or data set size, to more advanced ones that adapt to the performance metrics gathered in H_t.

3.4.1. Naive retraining policy: Periodic training

Periodic training is a naive class of retraining policies that focuses on system progress rather than model performance, ignoring H_t. Let q_t be a monotonically non-decreasing *progress measure* of the system, q_{t^*} its value at the last retraining time t^*, and N a fixed threshold. A periodic retraining policy retrains whenever q_t has advanced by at least N since the last cycle:

$$\pi_{per}(H_t, D_t) = \textbf{retrain} \iff q_t - q_{t^*} \geq N$$

We instantiate this class of policies in two ways: *(i) Time-based* and *(ii) Population-based.*

Definition 4 (*Time-based Retraining Policy*)**.** Let $q_t = t$, such that retraining occurs at fixed intervals T:

$$\pi_{time}(H_t, D_t) = \textbf{retrain} \iff t - t^* \geq T$$

Definition 5 (*Population-based Retraining Policy*)**.** Let $q_t = |D_t|$ be the cardinality of the aggregate dataset D_t, we define the population-based retraining policy π_{pop} as:

$$\pi_{pop}(H_t, D_t) = \textbf{retrain} \iff |D_t - D_{t^*}| \geq N$$

This type of retraining policy is straightforward to understand and implement; however, it is blind to actual model health: it may fire a retraining cycle even if the model is performing well or fail to fire promptly if performance degrades between scheduled cycles.

3.4.2. Adaptive retraining policies

In contrast, adaptive retraining policies implement a feedback loop by directly conditioning the retraining cycle on the performance metrics gathered in H_t. It is possible to define an adaptive retraining policy as follows.

Definition 6 (*Adaptive Retraining Policy*)**.** Let $(\Phi_t)_j$ be an aggregate performance metric computed at time t (e.g., accuracy or recall), θ_j be the minimum acceptable threshold for the jth performance metric. We define the adaptive retraining policy π_{ad} as:

$$\pi_{ad} : (H_t, D_t) = \textbf{retrain} \iff \exists j : (\Phi_t)_j < \theta_j$$

This policy can be extended, for example, by checking a sustained performance degradation over a window of consecutive monitorings, rather than reacting to each performance fluctuation. In this work, we implement three adaptive retraining policies: a performance-based policy that monitors accuracy degradation, and two policies based on well-known drift detection methods for machine learning, namely ADWIN [30] and DDM [31].

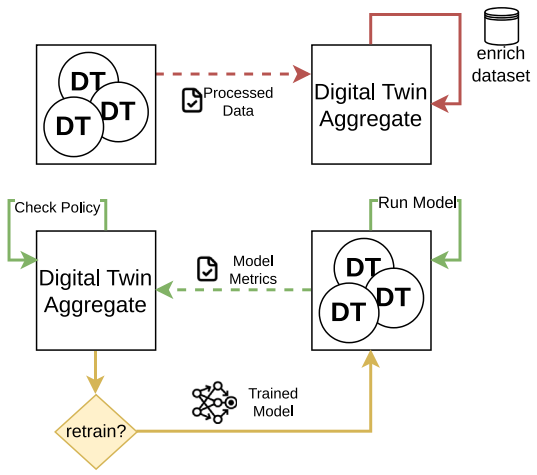

Fig. 2. Conceptual overview of the continuous lifecycle of the system. DTs share their state to the DTA, enriching its global view (top left–right). DTs also run their model prediction, gathering and sending metrics to the DTA (right-left). The DTA runs the retraining policy and can decide to retrain and share a new model (bottom left–right).

Definition 7 (*Performance-based Retraining Policy*)**.** Let $acc_t^i \in [0, 1]$ be the accuracy of the ith DT deployed model at time t, and let $acc_{t^*}^i$ be the accuracy of the model deployed to DT i at the last retraining time t^*. Let $\delta \in (0, 1)$ be the relative degradation tolerance and $p \in (0, 1]$ a *quorum threshold*. Each DT sets a decision flag $\phi_{t,c}^i$ when its accuracy has dropped, relative to deployment, by at least δ:

$$\phi_{t,c}^i = 1 \iff \frac{(acc_{t^*}^i - acc_t^i)}{acc_{t^*}^i} \geq \delta$$

We define the performance-based retraining policy π_{per} as:

$$\pi_{per} : (H_t, D_t) = \textbf{retrain} \iff \Phi_{t,c} = \frac{1}{n}\sum_{i=1}^{n} \phi_{t,c}^i \geq p$$

Definition 8 (*Drift-based Retraining Policy*)**.** Let $e_t^i \in 0, 1$ be the prediction error indicator of DT i's deployed model at time t, with $e_t^i = 1$ denoting model error, and let $e^i = (e_0^i, \ldots, e_t^i)$ be DT i's error stream. Let Δ be a drift detector that maps the error stream to a binary flag $\Delta(e^i) \in (\textbf{drift}, \textbf{no} - \textbf{drift})$, and let $p \in (0, 1]$ be a *quorum threshold*. Each DT sets its decision flag based on $\Delta(e^i)$:

$$\phi_{t,c}^i = 1 \iff \Delta(e^i) = \textbf{drift}$$

We define the drift-based adaptive retraining policy π_{drift} as:

$$\pi_{drift}(H_t, D_t) = \textbf{retrain} \iff \Phi_{t,c} = \frac{1}{n}\sum_{i=1}^{n} \phi_{t,c}^i \geq p$$

In Section 4, we instantiate Δ with two established detectors: DDM [31] and ADWIN [30].

3.5. Adaptive MLOps lifecycle

Once the policies and thresholds are defined, the three elements discussed: *(i) Abstract Architecture, (ii) Model Monitoring*, and *(iii) Retraining Policies* make up an MLOps lifecycle entirely managed by the DTA, without the need of additional entities and manual human intervention. Fig. 2 illustrates the lifecycle as a continuous process.

The cycle begins with data accumulation: for a certain time t, DTs shadow their physical counterparts and update their state, and the processed data flows to the DTA enriching the population-level data set. Concurrently, DTs run their current ML model against the incoming data, gathering their performance metrics ϕ_t and report it

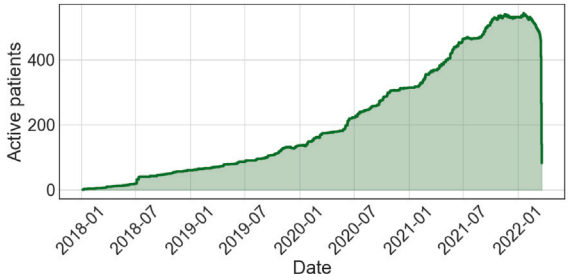

Fig. 3. Number of active patients over time in the T1DiabetesGranada dataset. The progressive growth of the monitored population highlights the dynamic nature of the case study and motivates the need for adaptive retraining policies managed at the DTA level.

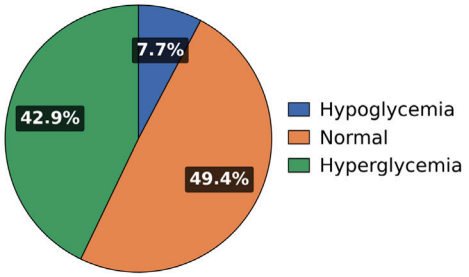

Fig. 4. Distribution of glycemic labels produced by the future-window labeling strategy described in Eq. (1).

back to the DTA, enabling continuous monitoring of the aggregate monitoring state Φ_t. At each monitoring step, the retraining policy π is evaluated: if it returns *wait*, the cycle continues with data accumulation and monitoring; if it returns **retrain**, the DTA initiates a retraining cycle, trains an updated model on the accumulated population data, and distributes it to all active DTs. The DTs adopt the new model, and the cycle restarts.

4. Experimental evaluation

To evaluate the proposed DTA framework, in this section, we present a healthcare case study of a longitudinal study on a growing population of patients and discuss how we simulated training ML models used by individual patient DTs over time to compare different retraining policies implemented at the aggregate level. The experiments are meant to demonstrate the feasibility of the proposed framework to encapsulate the MLOps pipeline and demonstrate that having an adaptive policy based on the reported performances evaluated by each individual DT can bring benefits to the overall quality of the ML model over time, compared to baseline methods.

4.1. Case study: Diabetic patient Digital Twins

Diabetes Mellitus (DM) is a chronic metabolic disease characterized by abnormal blood glucose regulation. This case study focuses on Type 1 Diabetes Mellitus (T1DM), where patients must continuously manage glucose levels through insulin administration, diet, and daily behavioral choices. Since glucose dynamics may rapidly evolve towards hypoglycemic or hyperglycemic conditions, T1DM represents a clinically relevant scenario for ML-enabled Human Digital Twins supporting proactive monitoring and early warning services.

We use the T1DiabetesGranada dataset [10], a longitudinal dataset collected from 736 T1DM patients over approximately four years. It provides patient-level glucose trajectories acquired through flash glucose monitoring devices, together with complementary demographic and clinical information. The dataset is particularly relevant for our evaluation because it captures both the individual temporal evolution of each patient and the progressive growth of the monitored population, resulting in a realistic setting where data availability, patient participation, and population composition change over time. These characteristics directly match the assumptions of our DTA-based architecture. Each patient can be represented by an individual HDT, which receives the patient-specific glucose stream and executes the deployed predictive model locally. The DTA, instead, maintains the aggregate view over the monitored population: it collects data and performance feedback from the HDTs, manages the population-level training set, evaluates retraining policies, and redistributes updated models when required. This separation is crucial because the dataset is not static. As shown in Fig. 3, patients are progressively introduced over time and contribute data for different temporal intervals. Consequently, both the active population and the amount of available glucose measurements evolve during the system lifecycle.

In this work, the predictive task is not formulated as direct glucose time-series forecasting. Instead, given the recent glucose history of a patient, the model predicts whether the patient will remain in the normal range or experience a hypo-/hyperglycemic event within a configurable future window, set to one hour in our experiments. The task therefore supports proactive alerting at the HDT level, while the DTA supervises model quality and adaptation across the evolving patient population.

4.2. Experimental setup

Learning task. As described in the previous section, all experiments are based on the T1DiabetesGranada dataset [10], which we preprocessed to cast the original longitudinal glucose monitoring data into the classification task considered in this work. In particular, the dataset was first split at patient level, ensuring that glucose measurements belonging to the same patient were never shared across different splits. Then, glucose measurements were labeled according to the clinical thresholds reported in [32], in order to distinguish normal glycemic conditions from hypo- and hyperglycemic events. More specifically, labels are assigned according to the future-window rule formalized in Eq. (1): a sample is labeled as `hyperglycemia` or `hypoglycemia` when the corresponding threshold is crossed at least once within the prediction window, and as `normal` when no glycemic event occurs.

$$y_t = \begin{cases} \text{hyperglycemia}, & \text{if } \exists k \in \{1,\dots,T\} : g_{t+k} \geq 180, \\ \text{hypoglycemia}, & \text{if } \exists k \in \{1,\dots,T\} : g_{t+k} < 70, \\ \text{normal}, & \text{otherwise.} \end{cases} \quad (1)$$

The resulting task consists in predicting, from the recent glucose history of a patient, whether the patient will remain in the normal range or experience a glycemic event within the future prediction window. Fig. 5 illustrates representative glucose trajectories crossing the adopted hypo- and hyperglycemia thresholds. Finally, since the resulting label distribution is imbalanced, as shown in Fig. 4, we adopted a class-weighted loss whose weights are computed according to the number of samples in each class, mitigating the bias towards majority classes during training.

Implementation details. All experiments were implemented in Python using the PyTorch framework [33]. In this evaluation, individual DTs were simulated within the experimental pipeline rather than deployed as independent distributed services. Accordingly, the experiments focus on the behavior of the retraining policies and on the model-level MLOps lifecycle, while aspects such as inter-DT communication, deployment latency, and runtime coordination overhead are left outside the scope of this work.

The full experimental code is publicly available under a permissive license at [34]. However, due to the license terms of the T1DiabetesGranada dataset, the repository only includes the source

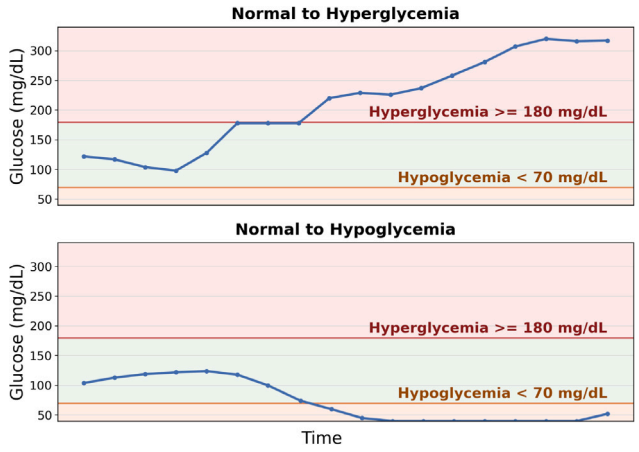

Fig. 5. Example glucose trajectories illustrating transitions from normal range to hyperglycemia and hypoglycemia according to the adopted clinical thresholds.

code and does not redistribute the data used in our experiments. Consequently, full reproducibility requires obtaining access to the dataset in accordance with its license before running the experimental pipeline. To reduce the impact of random initialization and data sampling effects, each configuration was evaluated over multiple random seeds. The results reported in the following section correspond to the average performance across 10 independent runs. This choice avoids selecting favorable seeds and provides a more statistically robust comparison among retraining strategies.

Finally, to ensure a fair comparison among retraining policies, all experiments use the same recurrent neural classifier. The model captures temporal patterns through a recurrent LSTM layer, which processes sliding input windows composed of the last 12 time steps with a stride of 6, and turns the final hidden representation into the predicted class through a fully connected head.

The model captures temporal patterns through a recurrent LSTM layer, which processes sliding input windows composed of the last 12 time steps with a stride of 6, and turns the final hidden representation into the predicted class through a fully connected head.

Retraining policies. Following the formulation introduced in Section 3.4, we evaluate three retraining policies with different levels of complexity. The goal is to compare simple, easy-to-implement policies against more adaptive strategies that exploit the monitoring capabilities provided by the DTA. In all cases, when a retraining condition is satisfied, the DTA retrains the model on the population-level data available up to that point and redistributes the updated model to the active HDTs. The first policy is a *time-based* retraining strategy. This policy represents the most naive baseline and triggers a retraining cycle periodically, independently of the number of new patients or of the observed model performance. The retrain interval is configurable; in our experiments, we set it to three months. This policy is simple and predictable, but it is blind to the actual state of the system: it may retrain the model even when performance is stable, or delay retraining despite an ongoing degradation. The second policy is *population-based*. In this case, retraining is triggered whenever the number of newly observed patients since the last retraining cycle exceeds a predefined threshold. This strategy explicitly accounts for the progressive growth of the monitored population and is therefore more closely aligned with the dynamic nature of the case study. However, similarly to the time-based policy, it does not directly monitor whether the deployed model is actually degrading.

Finally, we implement three adaptive retraining policies. The first one is *performance-based* and exploits the performance feedback collected by the DTA from active HDTs. A retraining cycle is triggered when the selected monitoring metric degrades beyond a predefined tolerance for a sufficiently large portion of the active DT population. In our experiments, the monitored metric is test accuracy: the policy triggers retraining when accuracy decreases by at least 10% for at least 20% of the active DTs. We further perform a sensitivity analysis (Table 1) by varying these thresholds and measuring their impact on the number of triggered retraining cycles; the selected configuration is retained as it provides a good trade-off between responsiveness and retraining frequency. In addition, we implement two drift-aware adaptive policies based on well-known machine learning drift detection methods, namely *ADWIN-based* and *DDM-based*. In these policies, each DT applies the corresponding detector to the stream of prediction errors generated by its deployed model and requests retraining when drift is detected. As for the performance-based policy, the DTA triggers retraining only when at least 20% of the active DTs request it.

4.3. Results

To evaluate the behavior of the proposed retraining policies, we monitored multiple metrics during training, validation, and inference. In particular, the test setting was designed to emulate the everyday use of the deployed model by active HDTs: at each timestep, the model is used to perform predictions on newly available patient data, and the resulting performance is reported back to the DTA. This allows us to evaluate not only the final predictive performance of each policy, but also its ability to maintain stable model quality over time. In addition, we monitored the number of triggered retraining cycles and the corresponding detection latency, in order to assess the responsiveness and operational behavior of each strategy. Finally, we also estimated the relative computational costs associated with each policy, providing a cost-aware perspective on the trade-off between model quality, retraining frequency, and deployment overhead.

The following paragraphs discuss these aspects separately and report the mean results over the different runs. When the results are intended to support direct comparisons of the improvements achieved by one policy over another, we additionally report variance and 95% confidence intervals, as in Table 1. Conversely, for plots whose purpose is to show that the observed trends are consistent and extend over the entire simulation horizon, such as Fig. 6, we report only mean values.

Predictive performance. Fig. 6 shows the evolution of test accuracy for the five retraining strategies. All adaptive policies (i.e., performance-based, ADWIN-based, and DDM-based), shown in Figs. 6(c) to 6(e), exhibits an initial transient phase in which the model undergoes some performance degradation. However, these degradations are detected by the DTA and addressed within a short time interval through targeted retraining. After this initial phase, the accuracy stabilizes and remains consistently high throughout the rest of the simulated lifecycle. This behavior suggests that conditioning retraining on actual performance feedback allows the system to react to model degradation without introducing unnecessary updates.

The time-based and population-based policies, shown in Figs. 6(a) and 6(b), also reach a stable accuracy regime in the long run. However, their accuracy curves are slightly more oscillatory, especially during the early and intermediate phases of the simulation. Since these policies do not directly observe model degradation, they can only compensate for performance drops when the corresponding temporal or population-size condition is eventually satisfied. As a consequence, they may either retrain when the model is still performing adequately or postpone retraining until the next scheduled or threshold-based trigger. This interpretation is further supported by the trend of test loss reported in Fig. 7, where all retraining policies exhibit a smooth decrease without divergent behavior, indicating that none of the configurations shows signs of overfitting and that the learning process remains stable.

Moreover, the recall trends reported in Fig. 8 further highlight the different behavior of the retraining policies evaluated. Time-based

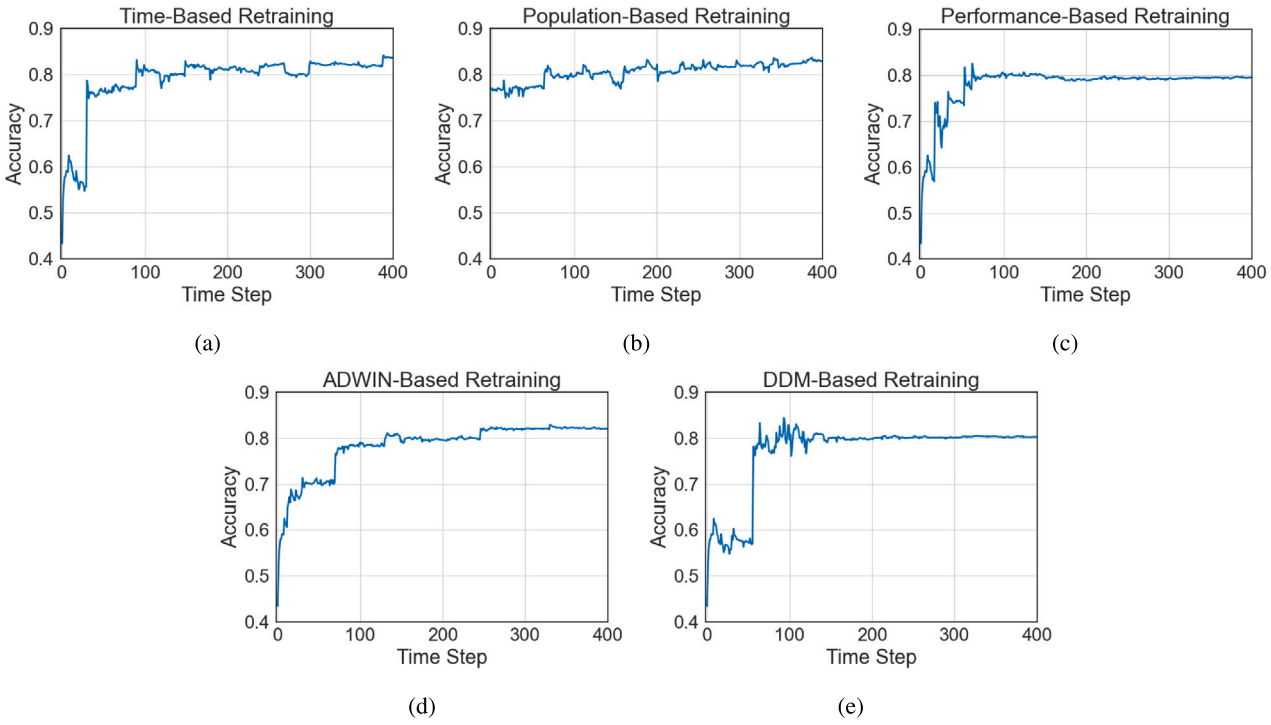

Fig. 6. Mean test accuracy over time for the evaluated retraining policies.

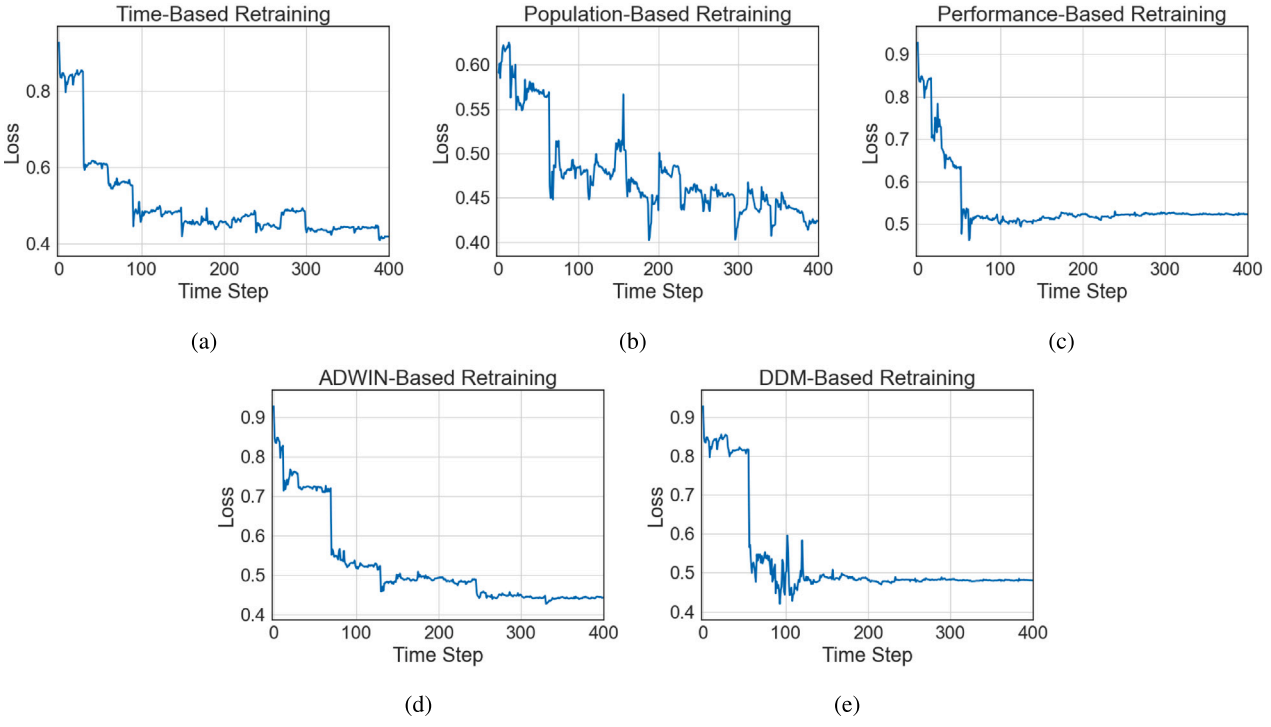

Fig. 7. Mean test loss over training steps for the evaluated retraining policies.

and population-based retraining exhibit more pronounced fluctuations, especially for the hypoglycemia class. This behavior can be attributed to the fact that these policies are not directly conditioned on model performance: retraining is triggered according to fixed temporal intervals or population-growth criteria, independently of whether the deployed model is actually degrading. As a consequence, a retraining cycle may replace a well-performing model with a new one whose decision boundaries differ from the previous version, potentially introducing unnecessary variability in the predictions. This effect is particularly visible for minority or harder-to-detect classes, where recall is more sensitive to small changes in the number of correctly identified positive samples. In contrast, the adaptive policies show a more stable recall profile over time. By triggering retraining only when a degradation or drift of the monitored performance metric is observed on a sufficiently large portion of the active DT population, this policy reduces unnecessary model updates and limits transient regressions introduced

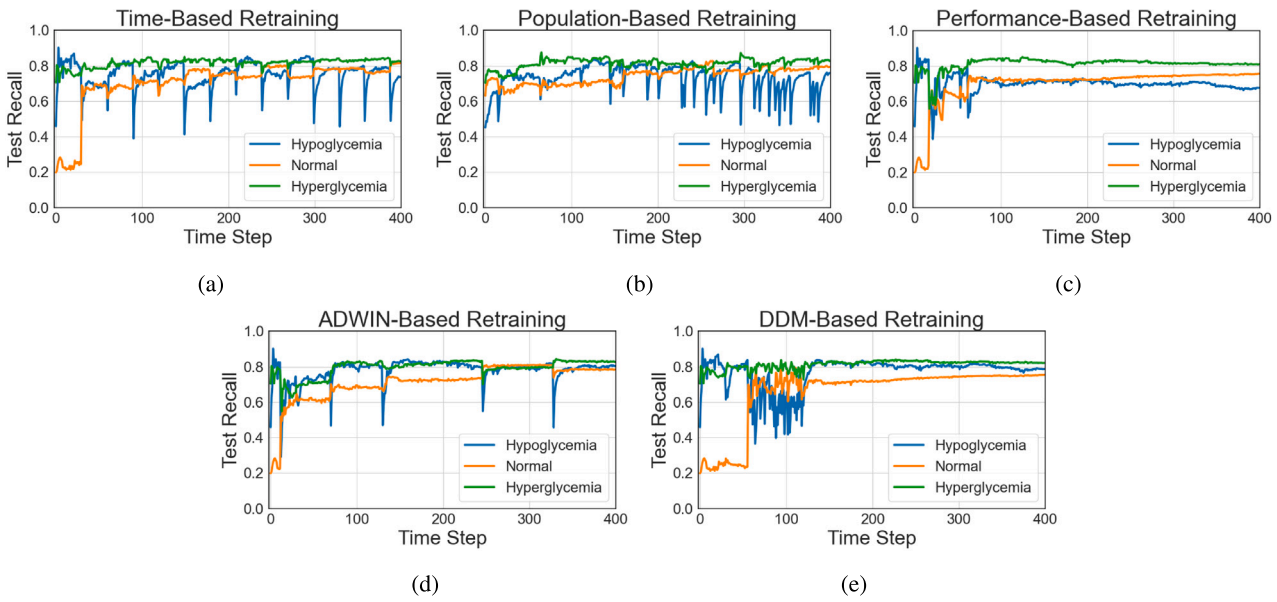

Fig. 8. Mean test recall over time for the evaluated retraining policies.

by non-performance-aware retraining. The resulting behavior suggests that performance and drift aware retraining provides a more conservative and targeted adaptation mechanism, maintaining model sensitivity more consistently across classes while limiting fluctuations.

These considerations are particularly relevant also from a clinical standpoint. The reliable detection of both hypoglycemic and hyperglycemic events is indeed essential, as timely intervention is required to prevent adverse health consequences. Therefore, errors affecting these pathological classes are of particular concern and should be minimized, considering the direct implications for patient safety and treatment management. Accordingly, retraining policies that maintain stable performance over the long term are preferable for clinical decision making.

Sensitivity analysis. We further analyze the sensitivity of the performance-based policy by varying the accuracy degradation threshold and the fraction of DTs required to trigger retraining. As reported in Table 1, the number of retraining cycles decreases as the policy becomes less aggressive: higher accuracy degradation thresholds and larger DTs fractions make retraining less frequent, while stricter settings increase responsiveness at the cost of additional retraining overhead. Importantly, the number of retraining cycles remains controlled across all configurations, showing that no threshold combination leads to an uncontrolled growth of retraining events or to unstable behavior that would negatively affect predictive performance. The configuration based on a 10% accuracy degradation threshold and a 20% DTs degradation threshold was therefore selected as a good compromise between timely adaptation and retraining frequency. To ensure a fair comparison across adaptive strategies, the same 20% population-level threshold was also used for the ADWIN-based and DDM-based policies, meaning that retraining is triggered only when at least 20% of the active DTs detect drift and request an update.

Retraining dynamics and detection latency. The difference among the various policies is further highlighted by the number of retraining cycles required by each policy. As reported in Fig. 9, the performance-based policy triggers only 7 retraining cycles over the four-year simulation, whereas the time-based and population-based policies require 16 and 25 retraining cycles, respectively. The ADWIN-based and DDM-based policies show a similar adaptive behavior, triggering 8 and 10 retraining cycles, respectively. Therefore, the adaptive strategies achieve a comparable, and more stable, test accuracy while substantially reducing the number of model updates with respect to fixed retraining schedules, especially the population-based policy. This reduction is particularly relevant in an MLOps setting, where retraining may involve non-negligible computational costs, validation effort, and deployment overhead.

Table 1
Sensitivity analysis of the performance-based policy under different accuracy and DTs degradation thresholds.

Accuracy degradation threshold	DT degradation threshold	Mean trainings	95% Confidence interval
5%	10%	20.4 ± 1.9	[19.4, 21.75]
	20%	15.2 ± 1.55	[14.1, 16.31]
	30%	11.9 ± 1.45	[10.86, 12.94]
10%	10%	8.4 ± 0.84	[7.79, 9.01]
	20%	7.1 ± 0.56	[6.69, 7.5]
	30%	5.0 ± 0.82	[4.42, 5.84]
15%	10%	6.1 ± 1.19	[5.25, 6.95]
	20%	4.1 ± 0.99	[3.89, 4.81]
	30%	2.2 ± 0.78	[1.64, 2.76]

We further analyzed the responsiveness of the adaptive policies by measuring the average delay between the beginning of a performance degradation and the triggering of a retraining cycle. Across the simulation, the performance-based policy shows an average response time of 26.6 days (as shown in Fig. 10). The ADWIN-based policy exhibits a comparable detection latency, while the DDM-based policy reacts more rapidly and with lower variability. The variability observed reflects the role of the retraining thresholds in controlling the sensitivity of the adaptive policies. Since retraining is triggered only when a sufficiently large degradation affects a minimum fraction of active DTs, lower thresholds would make the policies more reactive, reducing detection latency at the cost of more frequent retraining cycles. Conversely, stricter thresholds reduce unnecessary updates but may delay the detection of slower or less widespread degradation patterns, as suggested by the high-latency cases in the distribution. To obtain a comparable reaction time with a time-based policy, retraining would need to be scheduled approximately once per month. However, such a configuration would further increase the number of retraining cycles, leading to additional resource consumption without explicitly guaranteeing that each retraining is motivated by an actual degradation of model quality.

Finally, we note that the evaluated policies should not necessarily be regarded as mutually exclusive. In practical deployments, hybrid

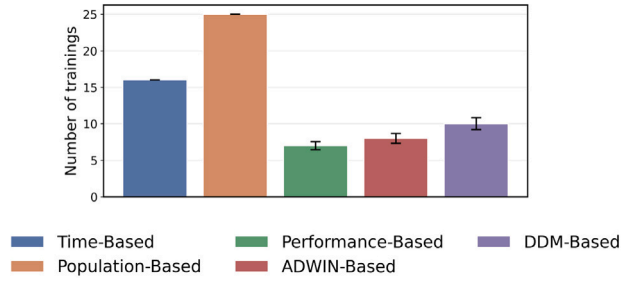

Fig. 9. Number of retraining cycles triggered by each policy during the four-year simulation.

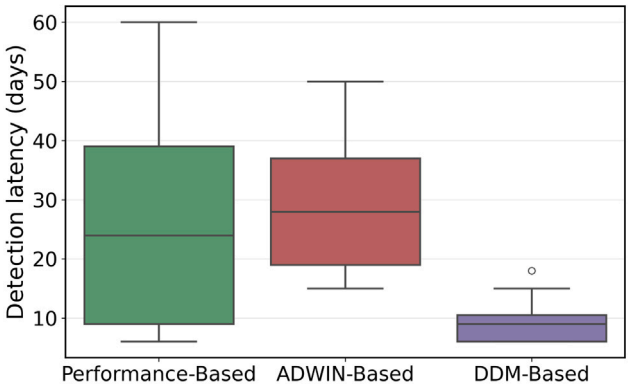

Fig. 10. Distribution of performance drift detection latency (in days) for the evaluated retraining policies.

retraining strategies could be adopted to combine the advantages of different triggers. For instance, adaptive policies could be complemented with a sparse time-based schedule (e.g., a yearly retraining cycle) or with a population-based policy, to guarantee occasional model refreshes even in the absence of detected degradation. In this way, even when the newly introduced patients do not immediately induce a measurable performance drop, their data can still be periodically incorporated into the training set to improve the model.

Estimated operational costs. We finally estimate the operational cost associated with each retraining policy. The total cost is not only determined by the number of retraining cycles, but also by when these cycles are triggered during the simulation. Indeed, retraining performed early in the deployment involves fewer accumulated samples and is therefore less expensive, whereas later retraining cycles are carried out on larger datasets and have a higher cost. To capture this effect, we use a normalized cost function defined as $cost(t) = C_{\text{fixed}} + C_{\text{row}} \cdot N(t)^{\alpha}$, where $N(t)$ is the number of available samples at retraining time t. We set $C_{\text{fixed}} = 1$, $C_{\text{row}} = 10^{-5}$, and $\alpha = 1$, so that costs are expressed in arbitrary normalized units and are used only for relative comparison across policies.

As shown in Fig. 11, blind policies such as the time-based and population-based strategies lead to the highest estimated costs. This happens because they trigger retraining according to fixed temporal or population-size conditions, without explicitly considering whether the model is actually experiencing degradation or drift. As a consequence, they may perform unnecessary and expensive model updates. In contrast, the adaptive policies substantially reduce the estimated cost by triggering retraining only when there is evidence of performance degradation or drift. This confirms that adaptive retraining can reduce deployment overhead while still preserving model quality over time.

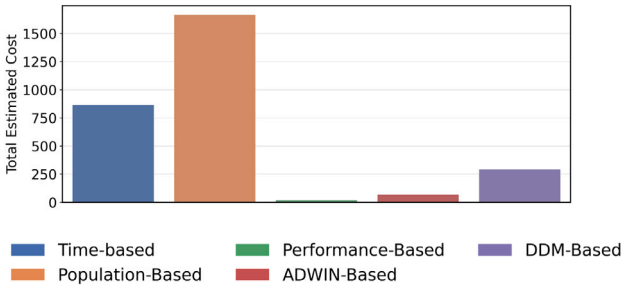

Fig. 11. Estimated normalized retraining cost for each policy over the four-year simulation.

5. Related works

As highlighted in Section 2, this work builds upon the convergence of several research directions towards effectively deploying ML models in healthcare-related scenarios, monitoring performances across independent DTs, while accounting for population-level dynamics through the concept of the DTA. The contribution focuses on establishing an architectural framework that encapsulates within the DT the responsibility of evaluating the ML model performance through the continuous acquisition of new data that can validate the inference output. At the same time, the responsibility of evaluating the overall system performance is encapsulated within the DTA. This is consistent with the expectation that the DTA can provide insights on a population of homogeneous DTs.

Despite being conceptually established since the early days of DTs, the engineering of DTAs is underexplored [35]. Microservice and Service-oriented architectures are among the main established architectural patterns for engineering DTs [36], making aggregation possible through established techniques. An example of this pattern is discussed in [24] where authors propose an architecture combining the data layers of different DTs to realize a hierarchical DTAs. We argue that this approach leans more towards DT composition [37], rather than aggregation, as this usually implies that the resulting composite DT models higher-level macro-scale physical assets through the composition of data sources that may be mediated by lower-level micro-scale DTs directly connected to the physical world [37]. For instance, a similar hierarchical composition of DTs is proposed in [38]. Second-level DTs are responsible for tracking KPIs alignment and energy consumption of a department in an industrial manufacturing scenario, acquiring data from individual machine-level DTs. In the healthcare domain, despite some surveys mentioning the possibility of employing DTAs [22,23] or mentioning DT-based population-level analysis [39], there are no architectural references to implement DTAs nor clearly defined responsibility boundaries.

Despite the lack of practical guidance in the engineering of DTAs, [35] proposes a general design framework to analyze and separate responsibilities in a system of DTs, using the architectural approach proposed in [24] to ground the framework with concrete application scenarios. Our proposal is coherent with the designed framework, as it follows the separation of concerns between the individual responsibilities of the DTs and the global performance monitoring service enacted by the DTA. The proposed DTA contributes to cementing the DTA abstraction as a valuable architectural component for DT-based systems. Aligned with its original definition [3], the proposed DTA does not qualify as a DT itself as it does not attempt to model an asset, but rather implements services – in this case MLOps services such as model performance evaluation and retraining scheduling – on top of a population of homogeneous DTs. We believe this contribution can further support exploring the engineering of DT based systems through the lens of the DTA abstraction.

For what concerns MLOps, a growing body of work has begun to exploit Digital Twins in MLOps contexts. While some use MLOps techniques only offline for the DT model development [40], some others incorporate MLOps processes within the DT itself: for instance, Kruschinski et al. [41] propose an MLOps framework for data-driven DT modeling, validated on a virtual test rig, and focused on the lifecycle of a single DT model. Lombardo et al. [42] present a multi-layer DT-based architecture that explicitly positions DTs as MLOps enablers and supports continual learning on top of Intelligent Location-Based Services. TWIN-ADAPT [43] addresses online adaptation in industrial DTs by detecting concept drift and updating anomaly classifiers within the DT framework.

These works confirm the viability of DT-based MLOps but operate primarily at the level of individual twins: model lifecycle management is performed per DT, without leveraging an aggregate-level view of a population of homogeneous instances. The concept of adopting a DTA for aligning and optimizing the MLOps workflow for multiple DTs of the same *type* is explored in [28], which extends the DTA architecture described in [24] with support for running MLOps workflows. Our approach is similar in handling the MLOps workflow of multiple DTs of the same *type* as defined in the paper and in the fact that the DTA encapsulates the triggers for the ML pipelines, as well as train the models on the data acquired from the whole population of DTs. Differently, though, to strengthen separation of concerns, our proposed DTA architecture does not support requesting the DTA to perform inference at the DT level. We assume each DT to be equipped with the capability of running the inference model.

Finally, our work is broadly positioned within the scope of "Edge-MLOps" [44], combining edge-inference with cloud-based MLOps pipelines and related to several efforts pairing MLOps in the healthcare domain for the dynamic evaluation of ML models. In this area, authors of [45], despite not referencing DTs, propose a two-tier architecture with edge inference and cloud-level monitoring and retraining that follows a similar rationale to the proposed approach. The Model-as-a-Service system component is implemented through the DTA in our proposal. Compared to previous works in the area, our frameworks connects the capabilities of DTs, acting as edge nodes that can automatically process incoming data, perform inference and evaluate model performance and potential drift, with the centralized role of the DTAs which can encapsulate retraining policies considering the reported signals of individual DTs.

6. Limitations, challenges and future works

This work is a first step towards a deeper integration of MLOps practices in DT-based systems. The core conceptual contribution, identifying DTAs as the abstraction that can capture model drifting analysis and proactive retraining and delivery of ML models, opens several interesting research directions. In this section, we comment on current limitations, potential challenges and research directions for future works in the area.

Although currently motivated and evaluated in a healthcare setting, the proposed approach is designed as domain-agnostic and likely applicable to all kinds of applications in which multiple DTs leverage the same model trained on the data of a population of homogeneous DTs for their local inferences. Nevertheless, the work makes several assumptions that may limit its applicability to other kinds of scenarios. First, the population of DTs should be *homogeneous*, i.e., they must share the same model and acquire the same data. Additionally, the core assumption here is that individual DTs may lack sufficient data to train independent models and may instead benefit from the generalization to a broader population. This implies that population-level patterns are relevant to improve performances of predictions at the local level. Finally, the current design of the solution considers the fact that DTs can share data with the DTA, acting as a central aggregation point for overall analysis of model performance trends to trigger retraining.

These assumptions may not hold in some systems, and hence make the approach not feasible as-is. Most notably, if the population distribution is highly heterogeneous, given the ambition of DTs to be an accurate representation of an *individual* entity, future works may explore the possibility of fine-tuning the DTA model trained on population data with local data to achieve a higher degree of personalization. This would open challenges for model performance assessments and possibly require a twofold evaluation of retraining, a local one at the DT level, and a global one at the DTA level. Additionally, in cases in which DTs are not allowed to directly share training data, Federated Learning [46] techniques might be adopted to collaboratively improve locally trained models. With this approach, the DTA could still be in charge of defining the overall retraining policy, but the training would no longer be centralized. To further break down the centralization aspect, decentralized federated-learning techniques could be used to let DTs self-organize within the federation [47,48].

The preliminary integration of MLOps techniques with DTs presented in this work is also currently limited in exploiting the DTs only for data acquisition, processing and model validation over time. In practice, DTs are typically envisioned as encapsulating multiple models. Future works may consider leveraging other models within the DTs to improve the ML model outcomes and retraining scheduling. In particular, the proposed framework can be extended towards physics-informed and hybrid modeling approaches, leveraging the fact that DTs naturally support the integration of data-driven and mechanistic representations. In this setting, the MLOps pipeline could exploit the underlying physical or physiological models embedded in the DT to constrain learning and ensure that model predictions remain consistent with known system dynamics and feasible operational ranges. Another relevant direction concerns the tighter integration of real-time sensor streams with simulation capabilities. Continuous data ingestion from distributed sensing infrastructures, combined with the simulation capabilities of DTs, could enable more proactive and anticipatory MLOps strategies, where model evaluation, validation, and retraining are driven not only by observed data but also by simulated future scenarios. This would further enhance robustness in highly dynamic environments and support earlier detection of model degradation.

For the current work, we limited our experimental evaluation to a cold-start training of the ML model, retrained from scratch at every iteration. Future works may consider evaluating whether continual learning could be combined with the proposed approach and improve the stability of the learning process as outlined in [42].

Finally, the retraining process and delivery of the model are straightforward in the current experimental settings. We deliberately avoided overloading the experiment setup with additional variables, but more complex retraining steps (e.g., testing different model architectures), or delivery strategies (e.g., conditionally update the model only on underperforming DTs) might create interesting dynamics in the overall behavior of the system. We leave the exploration of the effect of these variables for future works, as finding the optimal strategy for the specific case study is out of the scope of this paper which focuses instead on demonstrating the feasibility of the approach and comparing its features against baseline methods.

7. Conclusion

This work tackles the integration of MLOps practices in DT-based systems, repurposing the concept of DTA as a fundamental functional element of the MLOps pipeline. Namely, the work proposes that individual DTs instances can implement the acquisition, preprocessing, and performance monitoring steps of an MLOps workflow, while the DTA can aggregate data and performance indicators of the model being used by various DTs and hence encapsulate policies for retraining the model adapting to the model degradation on the population.

The proposed framework enhances the overall quality of DT-based systems by integrating model quality monitoring and retraining, which

would typically require manual intervention. At the same time, the framework enhances classic MLOps pipelines with an approach that separates responsibilities in a two-level architecture. Services (DTs) sharing the model to work on different data streams may independently assess the performances at the lower level. At the upper layer, a manager (DTA) evaluates global adaptive retraining policies to maximize the desired KPIs on the overall system.

The proposed approach is especially relevant in healthcare contexts, in which the longitudinal dimension of patient data and variability of population size can significantly alter the data distribution over time, requiring ML models adaptation. We hence validate the approach by simulating the evolution over time of a DT-based system in a glucose-monitoring case study for diabetic patients. The simulation leverages an approximately four-year longitudinal study dataset [10] to replay the evolution of the population of patients, each modeled by an individual DT collecting data over time, performing inferences, and monitoring the quality of the model. Results confirm that an adaptive policy – supported by the automatic feedback monitoring by each DT – implemented at the DTA level to assess retraining need over the performance of the whole population reaches comparable quality against baseline methods with less retraining over time and with a faster reaction time to model drift.

CRediT authorship contribution statement

Davide Domini: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Data curation, Conceptualization. **Leonardo Micelli:** Writing – original draft, Investigation, Conceptualization. **Samuele Burattini:** Writing – original draft, Investigation, Conceptualization. **Sara Montagna:** Writing – original draft, Methodology, Investigation, Funding acquisition, Conceptualization.

Declaration of competing interest

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

Acknowledgments

This work has been funded by the Italian Ministry of University and Research (MUR) programme “PRIN 2022 Scorrimento” - grant number 20222X4M4T - I-DASY - CUP: J53C24002660006

Data availability

The authors do not have permission to share data.

References

[1] G. Mallardi, L. Quaranta, F. Calefato, F. Lanubile, MLOps in the healthcare domain: a systematic literature review, in: D. Taibi, D. Smite (Eds.), Software Engineering and Advanced Applications, Springer Nature Switzerland, Cham, 2026, pp. 334–349.

[2] R. Minerva, G.M. Lee, N. Crespi, Digital twin in the IoT context: A survey on technical features, scenarios, and architectural models, Proc. IEEE 108 (10) (2020) 1785–1824.

[3] M. Grieves, J. Vickers, Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems, in: Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches, Springer International Publishing, Cham, 2017, pp. 85–113.

[4] E. Glaessgen, D. Stargel, The digital twin paradigm for future NASA and US air force vehicles, in: Proc. of the 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, 2012.

[5] E. Shahat, C.T. Hyun, C. Yeom, City digital twin potentials: A review and research agenda, Sustainability 13 (6) (2021).

[6] R. Saracco, Digital twins: Bridging physical space and cyberspace, Computer 52 (12) (2019) 58–64, http://dx.doi.org/10.1109/MC.2019.2942803.

[7] A. Ricci, A. Croatti, S. Montagna, Pervasive and connected digital twins—A vision for digital health, IEEE Internet Comput. 26 (5) (2022) 26–32, http://dx.doi.org/10.1109/MIC.2021.3052039.

[8] B. Björnsson, C. Borrebaeck, N. Elander, T. Gasslander, D.R. Gawel, M. Gustafsson, R. Jörnsten, E.J. Lee, X. Li, S. Lilja, D. Martínez-Enguita, A. Matussek, P. Sandström, S. Schäfer, M. Stenmarker, X.F. Sun, O. Sysoev, H. Zhang, M. Benson, on behalf of the Swedish Digital Twin Consortium, Digital twins to personalize medicine, Genome Med. 12 (1) (2019) 4.

[9] K. Sel, A. Hawkins-Daarud, A. Chaudhuri, D. Osman, A. Bahai, D. Paydarfar, K. Willcox, C. Chung, R. Jafari, Survey and perspective on verification, validation, and uncertainty quantification of digital twins for precision medicine, NPJ Digit. Med. 8 (1) (2025) 40, http://dx.doi.org/10.1038/s41746-025-01447-y.

[10] C. Rodriguez-Leon, M.D. Aviles-Perez, O. Banos, M. Quesada-Charneco, P.J. Lopez-Ibarra Lozano, C. Villalonga, M. Munoz-Torres, T1DiabetesGranada: a longitudinal multi-modal dataset of type 1 diabetes mellitus, Sci. Data 10 (1) (2023) 916.

[11] D.J. Hunter, C. Holmes, Where medical statistics meets artificial intelligence, N. Engl. J. Med. 389 (13) (2023) 1211–1219, http://dx.doi.org/10.1056/NEJMra2212850.

[12] P. Rajpurkar, E. Chen, O. Banerjee, E.J. Topol, AI in health and medicine, Nature Med. 28 (1) (2022) 31–38, http://dx.doi.org/10.1038/s41591-021-01614-0.

[13] S.E. Davis, R.A. Greevy, T.A. Lasko, C.G. Walsh, M.E. Matheny, Detection of calibration drift in clinical prediction models to inform model updating, J. Biomed. Inform. 112 (2020) 103611, http://dx.doi.org/10.1016/j.jbi.2020.103611.

[14] A. Rajagopal, S. Ayanian, A.J. Ryu, R. Qian, S.R. Legler, E.A. Peeler, M. Issa, T.J. Coons, K. Kawamoto, Machine learning operations in health care: A scoping review, Mayo Clin. Proc.: Digit. Health 2 (3) (2024) 421–437, http://dx.doi.org/10.1016/j.mcpdig.2024.06.009.

[15] S. Montagna, R. Stagni, G. Pierucci, A. Aceti, D.M. Cordelli, M.C. Bisi, Digital twins for monitoring neuromotor development in preterm infants: Conceptual framework and proof-of-concept study, J. Med. Syst. 49 (1) (2025) 143, http://dx.doi.org/10.1007/s10916-025-02252-6.

[16] L. Faust, P. Wilson, S. Asai, S. Fu, H. Liu, X. Ruan, C. Storlie, Considerations for quality control monitoring of machine learning models in clinical practice, JMIR Med. Inform. 12 (2024) e50437.

[17] C.J. Kelly, A. Karthikesalingam, M. Suleyman, G. Corrado, D. King, Key challenges for delivering clinical impact with artificial intelligence, BMC Med. 17 (1) (2019) 195, http://dx.doi.org/10.1186/s12916-019-1426-2.

[18] F. Bayram, B.S. Ahmed, Towards trustworthy machine learning in production: An overview of the robustness in MLOps approach, ACM Comput. Surv. 57 (5) (2025) http://dx.doi.org/10.1145/3708497.

[19] D. Kreuzberger, N. Kühl, S. Hirschl, Machine learning operations (MLOps): Overview, definition, and architecture, IEEE Access 11 (2023) 31866–31879, http://dx.doi.org/10.1109/ACCESS.2023.3262138.

[20] G. Mallardi, F. Calefato, L. Quaranta, F. Lanubile, An mlops approach for deploying machine learning models in healthcare systems, in: 2024 IEEE International Conference on Bioinformatics and Biomedicine, BIBM, 2024, pp. 6832–6837, http://dx.doi.org/10.1109/BIBM62325.2024.10822603.

[21] R. Zhang, F. Wang, J. Cai, Y. Wang, H. Guo, J. Zheng, Digital twin and its applications: A survey, Int. J. Adv. Manuf. Technol. 123 (11) (2022) 4123–4136, http://dx.doi.org/10.1007/s00170-022-10445-3.

[22] E. Katsoulakis, Q. Wang, H. Wu, L. Shahriyari, R. Fletcher, J. Liu, L. Achenie, H. Liu, P. Jackson, Y. Xiao, T. Syeda-Mahmood, R. Tuli, J. Deng, Digital twins for health: a scoping review, NPJ Digit. Med. 7 (1) (2024) 77, http://dx.doi.org/10.1038/s41746-024-01073-0.

[23] M.N. Kamel Boulos, P. Zhang, Digital twins: From personalised medicine to precision public health, J. Pers. Med. 11 (8) (2021) http://dx.doi.org/10.3390/jpm11080745, URL: https://www.mdpi.com/2075-4426/11/8/745.

[24] A.J.H. Redelinghuys, K. Kruger, A. Basson, A six-layer architecture for digital twins with aggregation, in: T. Borangiu, D. Trentesaux, P. Leitão, A. Giret Boggino, V. Botti (Eds.), Service Oriented, Holonic and Multi-Agent Manufacturing Systems for Industry of the Future, Springer International Publishing, Cham, 2020, pp. 171–182.

[25] A. Ricci, A. Croatti, S. Mariani, S. Montagna, M. Picone, Web of digital twins, ACM Trans. Internet Technol. 22 (4) (2022) http://dx.doi.org/10.1145/3507909.

[26] L. Micelli, S. Montagna, Human digital twin for healthcare applications: a white label digital twin implementation, in: Proceedings of the 2025 International Conference on Information Technology for Social Good, 2025, pp. 90–98.

[27] F. Abbasi, P. Brimont, C. Pruski, J.-S. Sottet, Understanding semantic drift in model driven digital twins, in: Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems, 2024, pp. 419–430.

[28] A.H. van Bruggen, K. Kruger, A.H. Basson, J. Grobler, An architecture to integrate digital twins and machine learning operations, in: T. Borangiu, D. Trentesaux, P. Leitão, L. Berrah, J.-F. Jimenez (Eds.), Service Oriented, Holonic and Multi-Agent Manufacturing Systems for Industry of the Future, Springer Nature Switzerland, Cham, 2024, pp. 3–14.

[29] A.L. Suárez-Cetrulo, D. Quintana, A. Cervantes, A survey on machine learning for recurring concept drifting data streams, Expert Syst. Appl. 213 (2023) 118934.

[30] A. Bifet, R. Gavaldà, Learning from time-changing data with adaptive windowing, in: Proceedings of the Seventh SIAM International Conference on Data Mining, April 26-28, 2007, Minneapolis, Minnesota, USA, SIAM, 2007, pp. 443–448, http://dx.doi.org/10.1137/1.9781611972771.42.

[31] J. Gama, P. Medas, G. Castillo, P.P. Rodrigues, Learning with drift detection, in: A.L.C. Bazzan, S. Labidi (Eds.), Advances in Artificial Intelligence - SBIA 2004, 17th Brazilian Symposium on Artificial Intelligence, São Luis, Maranhão, Brazil, September 29 - October 1, 2004, Proceedings, in: Lecture Notes in Computer Science, Springer, 2004, pp. 286–295, http://dx.doi.org/10.1007/978-3-540-28645-5_29.

[32] D. Govindaraju, S. Subbian, Event-triggered smart dual hormone artificial pancreas for patient-specific drug delivery, Sci. Rep. 15 (1) (2025) 32045.

[33] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch, 2017.

[34] D. Domini, domm99/experiments-2026-FGCS-dt-aggregates-for- adaptive-MLOps: 1.0.0, 2026, http://dx.doi.org/10.5281/zenodo.19821659.

[35] C. Human, A. Basson, K. Kruger, A design framework for a system of digital twins and services, Comput. Ind. 144 (2023) 103796, http://dx.doi.org/10.1016/j.compind.2022.103796, URL: https://www.sciencedirect.com/science/article/pii/S0166361522001920.

[36] E. Ferko, A. Bucaioni, M. Behnam, Architecting digital twins, IEEE Access 10 (2022) 50335–50350, http://dx.doi.org/10.1109/ACCESS.2022.3172964.

[37] W. Jia, W. Wang, Z. Zhang, From simple digital twin to complex digital twin part I: A novel modeling method for multi-scale and multi-scenario digital twin, Adv. Eng. Inform. 53 (2022) 101706, http://dx.doi.org/10.1016/j.aei.2022.101706, URL: https://www.sciencedirect.com/science/article/pii/S1474034622001641.

[38] M. Martinelli, J. Zhang, A.-K. Splettstoßer, M. Picone, M. Lippi, A. Wortmann, Hierarchical digital twin ecosystem for industrial manufacturing scenarios, in: 2024 50th Euromicro Conference on Software Engineering and Advanced Applications, SEAA, 2024, pp. 56–63, http://dx.doi.org/10.1109/SEAA64295.2024.00018.

[39] A. Vallée, Digital twin for healthcare systems, Front. Digit. Health Volume 5 - 2023 (2023) http://dx.doi.org/10.3389/fdgth.2023.1253050, URL: https://www.frontiersin.org/journals/digital-health/articles/10.3389/fdgth.2023.1253050.

[40] T.Y. Fujii, V.T. Hayashi, R. Arakaki, W.V. Ruggiero, R. Bulla, F.H. Hayashi, K.A. Khalil, A digital twin architecture model applied with mlops techniques to improve short-term energy consumption prediction, Machines 10 (1) (2022) http://dx.doi.org/10.3390/machines10010023, URL: https://www.mdpi.com/2075-1702/10/1/23.

[41] D. Kruschinski, D.T. Ngassam, U. Durak, S. Hartmann, An MLOps framework to data-driven modelling of digital twins with an application to virtual test rigs, in: Advances in Conceptual Modeling, Springer Nature Switzerland, Cham, 2025, pp. 71–86.

[42] G. Lombardo, M. Picone, M. Mamei, M. Mordonini, A. Poggi, Digital twin for continual learning in location based services, Eng. Appl. Artif. Intell. 127 (2024) 107203.

[43] R. Gupta, B. Tian, Y. Wang, K. Nahrstedt, TWIN-ADAPT: continuous learning for digital twin-enabled online anomaly classification in iot-driven smart labs, Futur. Internet 16 (7) (2024) 239.

[44] E. Raj, D. Buffoni, M. Westerlund, K. Ahola, Edge MLOps: An automation framework for AIoT applications, in: 2021 IEEE International Conference on Cloud Engineering, IC2E, 2021, pp. 191–200, http://dx.doi.org/10.1109/IC2E52221.2021.00034.

[45] A.S. Ilic, M. Ilic, N. Stojanovic, D. Stojanovic, Adaptive healthcare monitoring through drift-aware edge-cloud intelligence, Futur. Internet 18 (3) (2026) http://dx.doi.org/10.3390/fi18030156, URL: https://www.mdpi.com/1999-5903/18/3/156.

[46] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: A. Singh, X.J. Zhu (Eds.), Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA, in: Proceedings of Machine Learning Research, PMLR, 2017, pp. 1273–1282, URL: http://proceedings.mlr.press/v54/mcmahan17a.html.

[47] D. Domini, G. Aguzzi, L. Esterle, M. Viroli, FBFL: A field-based coordination approach for data heterogeneity in federated learning, Log. Methods Comput. Sci. 22 (1) (2026) http://dx.doi.org/10.46298/LMCS-22(1:19)2026.

[48] D. Domini, N. Farabegoli, G. Aguzzi, M. Viroli, L. Esterle, Decentralized proximity-aware clustering for collective self-federated learning, Internet Things 35 (2026) 101841, http://dx.doi.org/10.1016/j.iot.2025.101841, URL: https://www.sciencedirect.com/science/article/pii/S2542660525003555.