RESEARCH-ARTICLE

Context-aware Drift Detection for Quality-aware Data-Driven Smart City Digital Twins

LUCA SERFILIPPI, University of Bologna, Bologna, BO, Italy

ARMIR BUJARI, University of Bologna, Bologna, BO, Italy

ANTONIO CORRADI, University of Bologna, Bologna, BO, Italy

Published: 09 December 2025

GoodIT '25: International Conference on Information Technology for Social Good
September 3 - 5, 2025
Antwerp, Belgium

Conference Sponsors:
SIGCAS

https://doi.org/10.1145/3748699.3749824
ISBN: 9798400720895

Context-aware Drift Detection for Quality-aware Data-Driven Smart City Digital Twins

Luca Serfilippi*
University of Bologna
Bologna, Italy
luca.serfilippi@unibo.it

Armir Bujari*
Università di Bologna
Bologna, Italy
armir.bujari@unibo.it

Antonio Corradi*
University of Bologna
Bologna, Italy
antonio.corradi@unibo.it

Abstract

As urban environments become more complex, Smart Cities will increasingly depend on the integration of Internet of Things (IoT) devices and Digital Twin (DT) technologies to enable real-time monitoring, simulation, and predictive analytics, to ultimately improve quality of life. Data-driven approaches play a crucial role in optimizing city operations, but their effectiveness is hampered by data drift, shifts in data distributions over time that can degrade model performance. Frequent model retraining is a common solution, but it can be ineffective or lead to high computational costs. Alternatively, drift-driven approaches are often vulnerable to false detections caused by noisy IoT data, hardware failures, or cyber threats. To address that problem, we propose a context-aware drift detection algorithm that takes advantage of local correlations between data sources to detect actual real drifts while avoiding false positives. Furthermore, we present a decentralized layered architecture for embedding drift detection within the Smart City ecosystem and evaluate our approach by using a real-world dataset. Our approach demonstrates the viability of context-aware distributed drift detection, enhancing the reliability and efficiency of ML-driven Smart City applications.

CCS Concepts

• **Computing methodologies** → *Online learning settings*; • **Information systems** → *Stream management*; • **Computer systems organization** → *Embedded and cyber-physical systems.*

Keywords

Digital Twin, Drift Detection, Smart City, Sustainability

ACM Reference Format:
Luca Serfilippi, Armir Bujari, and Antonio Corradi. 2025. Context-aware Drift Detection for Quality-aware Data-Driven Smart City Digital Twins. In *International Conference on Information Technology for Social Good (GoodIT '25), September 03–05, 2025, Antwerp, Belgium.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3748699.3749824

1 Introduction

In an era of rapid urbanization, the concept of a Smart City (SC) emerges as a transformative approach to address the multifaceted challenges of modern urban environments [2]. As cities expand, effective management of their development becomes essential to ensure sustainability and resilience. Part of this technological growth is made possible by the diffusion of Internet of Things (IoT), which can provide a network of interconnected devices, sensors and systems embedded in the urban landscape, allowing the generation of large amounts of real-time data to be collected and handled [23].

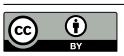
However, transforming raw data into significant information requires innovative frameworks that can assist city management. This is where Digital Twins (DTs) play a crucial role in providing virtual replicas of physical assets, processes, and systems, to enable real-time monitoring, simulation, and predictive analytics [10]. DTs facilitate data-driven decision-making, to improve both the efficiency and responsiveness of city operations.

A fundamental component of the DT could be *Machine Learning (ML)* [5, 11, 20, 21], a data-driven modeling approach that can learn patterns from data to make predictions or decisions, empowering both simulation and decision-making in response to the key challenges facing SC. ML-driven approaches have been proven to be effective in optimizing traffic flow, predicting air quality, managing disaster response, and improving face recognition [7]. However, maintaining ML models in production environments poses significant challenges, especially in dynamic and evolving contexts such as smart cities.

A critical issue in ML is model drift [4], intended as the change in data distributions and relationships over time that can cause a degradation of model performance. This is particularly relevant in SC, where environment, social, and technical variables change constantly. Correcting model drift typically requires either model retraining or updating [6, 16, 18, 19]. However, since frequent retraining is computationally expensive and resource-intensive, it is crucial to correctly and precisely identify real drifts to avoid false drift detections and carefully balance the trade-off between retraining costs and model performance [14, 17].

Furthermore, information from IoT devices is subject to noise, hardware malfunctions, and malicious attacks [9, 12, 15], all of which can be detected as drifts, that we will call *anomalous drifts*. However, these false drifts must not trigger retraining to avoid unnecessary costs. Moreover, we want to model the real city/phenomenon and not the errors induced by the sensing pipeline. The challenge lies in separating *anomalous drift* from *systematic drift*, the latter indicating true changes in the observed system. Our approach classifies these drifts by examining correlations between neighboring data sources.

To ensure robust and cost-effective model maintenance, we have designed a multi-layered architecture, conceptually organized into three distinct layers. First, the Data Layer serves as the foundation, housing the functional components responsible for interfacing with



GoodIT '25, Antwerp, Belgium

ACM ISBN 979-8-4007-2089-5/25/09
https://doi.org/10.1145/3748699.3749824

sensors, managing data processing pipelines, and handling all data storage. Building upon this, the Service Layer provides essential backend services for application developers and, crucially, is where our concept-drift detection mechanisms reside. Finally, the Application Layer sits at the top, enabling the development of Digital Twin (DT) applications that seamlessly leverage the infrastructure services provided by the underlying layers. Without loss of generality, we assess our proposed concept-drift algorithmic approach relying on a real dataset collected from the city of Bologna [1]. Results show that our proposal compared to other standard metods can reduce the costs of maintainence of ML models of DTs. Moreover, introducing syntetic anomalies in the dataset we show the accuracy of the context-aware drift detection in different scenarios.

The article is organized as follows: Section 2 provides a concise background on the concepts and technological ecosystem adopted in this work. Section 3 provides an overview of related works. Section 4 describes the algorithm used for context-aware drift detection, along with a potential concrete implementation of the proposed algorithm, followed by the proposed architecture for embedding the proposal in the smart city context. Section 5 starts by describing the framework and the data for testing the proposed solution, followed by the results showing how our solution compares to the baseline algorithms. Finally, Section 6 draws the conclusions, delineating some future work.

2 Background

This section sets the stage for our proposed solution by first providing some background literature on the concept of Smart City Digital Twin, then reviewing common correction strategies for ML model maintenance.

2.1 Smart City Digital Twin

For the purposes of this work, we consider SC Digital Twin not as a single monolithic DT, but as a dynamic ecosystem populated by DTs [10]. This ecosystem is founded upon a data layer characterized by diverse and heterogeneous data streams originating from a wide array of city sensors and IoT devices [23]. On top of this layer, a series of interconnected applications are strategically deployed across interconnected computational resources, spanning from (far)edge devices to centralized cloud platforms.

Each application within this ecosystem is designed to perform various tasks under the umbrella of urban management. A key characteristic is that these distributed applications are empowered by ML models [5, 11, 20, 21]. A general application typically employs an ML model tailored to a specific urban phenomenon, e.g., traffic optimization, air quality prediction [7] etc. Each such specialized model operates on a relevant subset of the overall heterogeneous city data, selecting the specific data streams relevant to its designated task.

These models provide core intelligence, enabling applications to learn from their specific data subset, identify complex patterns, make accurate predictions, and automate decision-making processes. Consequently, the performance and reliability of the urban services managed by these specialized Digital Twin instances are heavily dependent on the effectiveness of their underlying, focused ML models. However, these ML models, being data-driven and specialized, need to be actively maintained throughout their lifecycle to ensure continued accuracy and relevance in a constantly changing urban environment [4, 16, 19]. They collect their specific data streams to serve a dual purpose: (i) create a global view to build the initial ML models (training) and update them over time (retraining), and (ii) to process it for real-time decision-making (inference). Given this architecture of distributed, specialized ML models, each relying on specific evolving data subsets, drift detection emerges not just as a utility, but as a fundamental system service. This service can be used regardless of the specialized nature of the applications, providing a common utility to maintain the quality of the model and operational effectiveness within the dynamic Smart City Digital Twin ecosystem.

2.2 Drift Correction Strategies

Two common strategies for maintaining the accuracy of ML models in dynamic environments are periodic retraining and drift-driven retraining [14].

Periodic Retraining: This is a straightforward approach where models are retrained at regular, predetermined intervals, e.g., daily, weekly, monthly. This solution is easy to implement and schedule, ensuring that the models are regularly updated with recent data. However, it often leads to unnecessary retraining cycles when no significant changes have occurred in the modeled system.

Drift-Driven Retraining: In this strategy, model retraining is triggered only when a drift detection mechanism signals a significant change in the data [4]. The drift calculation is also performed periodically, but its cost with respect to retraining is negligible. This solution reduces consumption of resources on model retraining. However, the effectiveness of this strategy relies on the accuracy of the drift detection algorithm, False positive detections, which can be triggered by noisy IoT data, sensor malfunctions, or even cyber threats[9, 12, 15], can lead to unnecessary retraining, nullifying some of the cost benefits.

Choosing an appropriate strategy involves balancing the computational cost of retraining with the potential cost or penalty incurred from performance degradation due to unreported or unaddressed drift [6, 14, 18, 19]. The challenge of false positives in drift detection, particularly in noisy IoT environments [9, 12, 15], motivates research to find more robust detection methods, such as the context-based approach explored in this work.

3 Related Work

Research on maintaining ML models in dynamic environments like Smart Cities has yielded a variety of drift detection algorithms designed to identify changes in data properties over time. These approaches range from statistical methods that formally compare data distributions between different time windows (using tests such as Kolmogorov-Smirnov or Chi square tests, or divergence measures such as KL divergence), to techniques that monitor specific statistical properties of the data stream, and even machine learning-based methods that attempt to classify whether new data points belong to a drifted distribution [6, 16, 18, 19].

A typical action after a drift detection is model retraining, and optimization is crucial due to its computational cost and the trade-off against performance degradation penalties. Addressing this,

the authors of [17] investigate cost-aware retraining strategies aimed at balancing these factors. Similarly, another study explicitly models the maintenance problem by analyzing the trade-off between retraining costs and penalty costs incurred from performance degradation (e.g. false positives/negatives) [14]. This work uses probabilistic models to compare different retraining strategies (periodic vs. detector-based) and determine optimal policies that minimize the average long-term cost.

A significant challenge in IoT-based systems is the high rate of false positives in drift detection. These can be triggered by sensor faults rather than true changes in the underlying process. Darvishi *et al.* in [9] propose a modular data-driven architecture specifically for sensor-fault detection, isolation, and accommodation within Digital Twins, highlighting the need to handle sensor unreliability. Furthermore, IoT systems are vulnerable to attacks that might mimic drift. The authors in [15] provides a critical review of intrusion detection systems and techniques in IoT, while Haque *et al.* [12] offer a systematic review focused on data-driven attack detection trends in IoT, underscoring the security challenges that can complicate drift detection.

The use of context represents a promising direction for improving drift detection accuracy, and other research has also explored context-aware drift detection from a different perspective. As an example, the authors in [8] propose a general statistical framework based on conditional distributional treatment effects, borrowing from causal inference. Their approach aims to detect drift conditional on a changing context, allowing drifts when the context changes.

In this work, we introduce contextual information to drift detection with the aim of distinguishing between localized sensor issues and genuine/real data shifts. The goal is to analyze data streams not in isolation but related to neighboring sensors, identified by relying on a notion of distance, increasing the reliability of drift detection and preventing unnecessary model updates triggered by isolated faults. Additionally, we provide researchers and practitioners with a comprehensive, ready-to-use recipe, consisting of a layered system architecture within which context drift detection services are embedded.

4 Drift Detection in the Smart City

In this section, we introduce the algorithm proposed to detect and classify data drifts, followed by an architecture designed to embed the algorithm in the smart city context.

4.1 Context-aware Drift Detection

Drift detection involves monitoring temporal changes in the data distribution of sensors. Without loss of generality, let us consider a traffic management scenario where sensors record the count of vehicles passing through an area (n). Consider, for example, a sensor that typically records a daily count of $n = 500$ vehicles on a particular road. If this sensor suddenly starts reporting a null count of ($n = 0$), a significant data drift is evident. Although standard drift detection algorithms can easily identify this shift, they cannot ascertain its type.

This observed data deviation could arise from two distinct scenarios, each requiring a different remediation strategy:

- A *Systematic drift*, such as an actual road closure. This represents a genuine change in the monitored environment, potentially invalidating existing predictive models and necessitating retraining with new representative data.
- An *Anomalous drift*, such as a sensor malfunction or a localized data transmission error. This indicates a problem with the data source itself, rather than a change in the underlying system. The appropriate response is to investigate and rectify the sensor issue; the existing model, once provided with the correct data or by excluding the fault period, may still be valid.

Identifying these two types of drift requires contextual information beyond the readings of the single affected sensor. In SC, data from different sensors is clearly locally correlated. Extending our example, if another car counter on an adjacent road also registers a significant drop in traffic, the initial drift is likely *systematic* (e.g., the road closure). Conversely, if the neighboring sensor remains consistent with typical patterns, the drift in the first sensor is more likely *anomalous*, pointing towards an isolated sensor issue.

Therefore, we introduce a context-aware drift classification algorithm. Once a drift is detected for a specific sensor, this algorithm leverages information from its neighboring sensors to determine if the drift is *systematic* (signalling a true change in the underlying system) or *anomalous* (likely due to sensor-specific problems like faults, noise, or localized, non-representative events).

The classification process is formalized as follows: let S denote the set of sensors deployed in the monitored space. For each sensor s_i, we create a set of neighbors, computed as:

$$N(s_i) = \{\forall s_j \in S \setminus \{s_i\} : C(s_i, s_j)\} \tag{1}$$

where $C(x, y)$ is a grouping function that evaluates the affinity among nodes. As an example, one might require grouping co-located nodes using a distance metric, or employ more advanced grouping strategies based on data cross-entropy [22].

Each sensor $s_i \in S$ produces a data stream, which is segmented into discrete periods $s_i(\tau_j)$ where τ_j is the time span. These periods could, for example, align with the intervals of a periodic retraining schedule. Let s_i denote a sensor for which a primary drift detection mechanism has signalled a drift, as defined by the following:

$$drift(s_i, \tau_j) = drift(s_i(\tau_j), s_i(\tau_{j-1})) \tag{2}$$

where the $drift$ function can be a statistical test that compares a recent data distribution with a reference distribution. Given the set of drifting neighbors noted as:

$$D(s_i) = \{s_k \in N(s_i) : drift(s_k, \tau_j)\} \tag{3}$$

and upon calculating the ratio of drifted neighbouring sensors given by:

$$r(s_i) = \frac{|D(s_i)|}{|N(s_i)|} \tag{4}$$

we are able to classify the contextual drift based on a parametric threshold h where $0 < h < 1$,

$$contextualDrift(s_i) = \begin{cases} Systematic & \text{if } r(s_i) > h \\ Anomalous & \text{if } r(s_i) < h \end{cases} \tag{5}$$

h indicates the ratio of neighbors that have to drift concurrently to label the contextual drift as *systematic*. The threshold h can be tuned based on the accuracy of the grouping function in classifying highly correlated neighbors. A high value of h restricts the classification of *systematic* drifts to cases where most sensors are strongly correlated, typically requiring correlations above 0.5, similar to a majority vote. In contrast, a low h allows the detection of *systematic* drifts even when many neighboring sensors are not drifting, with $h = 0$ reducing the method to a standard drift-based approach.

4.2 Context-aware Drift Detection in the Digital Twin Architecture

To embed our approach to drift classification in the SC, we propose the insertion of context-aware drift detection within the digital twin architecture at the Service Layer, positioned between the data sources (Data Layer) and the digital twin applications (Application Layer) as depicted in Figure 1.

- The *Data Layer* collects IoT data from heterogeneous sensors in the urban environment via communication protocols commonly found in IoT ecosystems, such as MQTT, ReST APIs, OPC-UA, and CoAP. Furthermore, this layer is responsible for essential data processing, such as data enrichment, which involves augmenting raw sensor data with additional contextual information (e.g., timestamps, sensor location); preprocessing steps, involves filtering, normalization, and transformation. Aggregation, which combines data from multiple sources or over time periods to provide summarized views.
- The *Service Layer* provides system-level functions, such as context-aware message-oriented middleware and drift detection services. We view drift detection as a fundamental system service, designed to monitor the data layer and enrich the data stream with semantic labels, identifying and classifying drifts as *systematic* or *anomalous*. This information is then made available to the application layer via standard APIs.
- The *Application Layer* comprises various intelligent vertical DTs that subscribe to streams of control/IoT data and independently decide whether to retrain models based on contextual drift signals. This offers several system goods: reusability of drift detection across applications, scalability through distributed processing, extensibility for integrating new sensors or applications, and resilience against isolated drifts, improving the precision of retraining decisions.

In our approach to maintain the DT, drift detection at the Service layer serves as a middleware component, orchestrating the flow of information and facilitating distributed intelligence. It moves drift detection computations from individual applications to the infrastructure, enabling cooperative calculations for classifying drifts as *systematic* or *anomalous* and reducing redundant processing when multiple applications utilize the same data sources. Each Sensor data stream is associated with a drift detection component that augments the raw data with drift information. Central to the communication within the service layer is a Message-Oriented Middleware (MoM) backbone. The MoM facilitates the inter-component communication essential for our drift classification algorithm. Drift detection components themselves subscribe to topics from their designated neighbors. This enables them to share their drifting state with other relevant components in the infrastructure, compare these states, and collectively decide on the nature of a detected drift (*systematic* or *anomalous*) based on local consensus. This distributed approach also enables a rapid integration of new sensors into the city, as they can easily publish their data and subscribe to relevant contextual information, enhancing the labeling of already existing sensors used by various applications.

While this distributed architecture introduces communication and computation overhead, it is a deliberate trade-off designed to achieve greater accuracy and reduce costly, unnecessary model retraining. The communication overhead, managed by the MoM, primarily consists of lightweight status messages (e.g., drift detected/not detected) rather than large data payloads. This overhead is further managed by restricting the communication at the local level, performing neighbor discovery periodically at low frequencies. The computational overhead of running statistical tests (like the KS-test) within each component is negligible compared to the cost of full model retraining. By distributing this lightweight computation, the architecture avoids a central bottleneck and enhances scalability. Therefore, the marginal increase in system overhead is justified by the significant savings in computational resources and improved model reliability achieved by avoiding false-positive-triggered retraining.

4.3 Implementation of a Distance-based Context-Drift Service

We now present an implementation of the drift detection service component shown in Figure 2, describing its properties. We can distinguish two phases: the configuration phase and the operation. During the configuration phase, the component determines its operational environment, by creating its own drift topic on the Contextual Drift MoM, which includes metadata indicating its precise position within the city. Subsequently, it scans the other topics comparing its position with the others to create its neighborhood (Eq. 1) using the following comparing function.

$$C(s_i, s_j) = distance(pos(s_i), pos(s_j)) < radius \tag{6}$$

These discovered topics are identified as its neighbours $N(s_i)$, subscribing to their channels. A visualization of the neighborhood is shown in Figure 3.

After configuration, the component enters the operation phase, where it performs periodic tasks to monitor and assess the sensor data:

- *Data Collection*: the component continuously collects data from the associated sensor feed by Data Layer processes $(s_i(\tau_j))$.
- *Drift Detection*: analyzes the incoming data to identify potential drifts, which refer to significant deviations in a period of sensor readings (Eq. 2).
- *Neighbour Monitoring*: the component also listens for drift notifications from its neighboring nodes within the defined radius and checks if new neighbors have been introduced in the city.

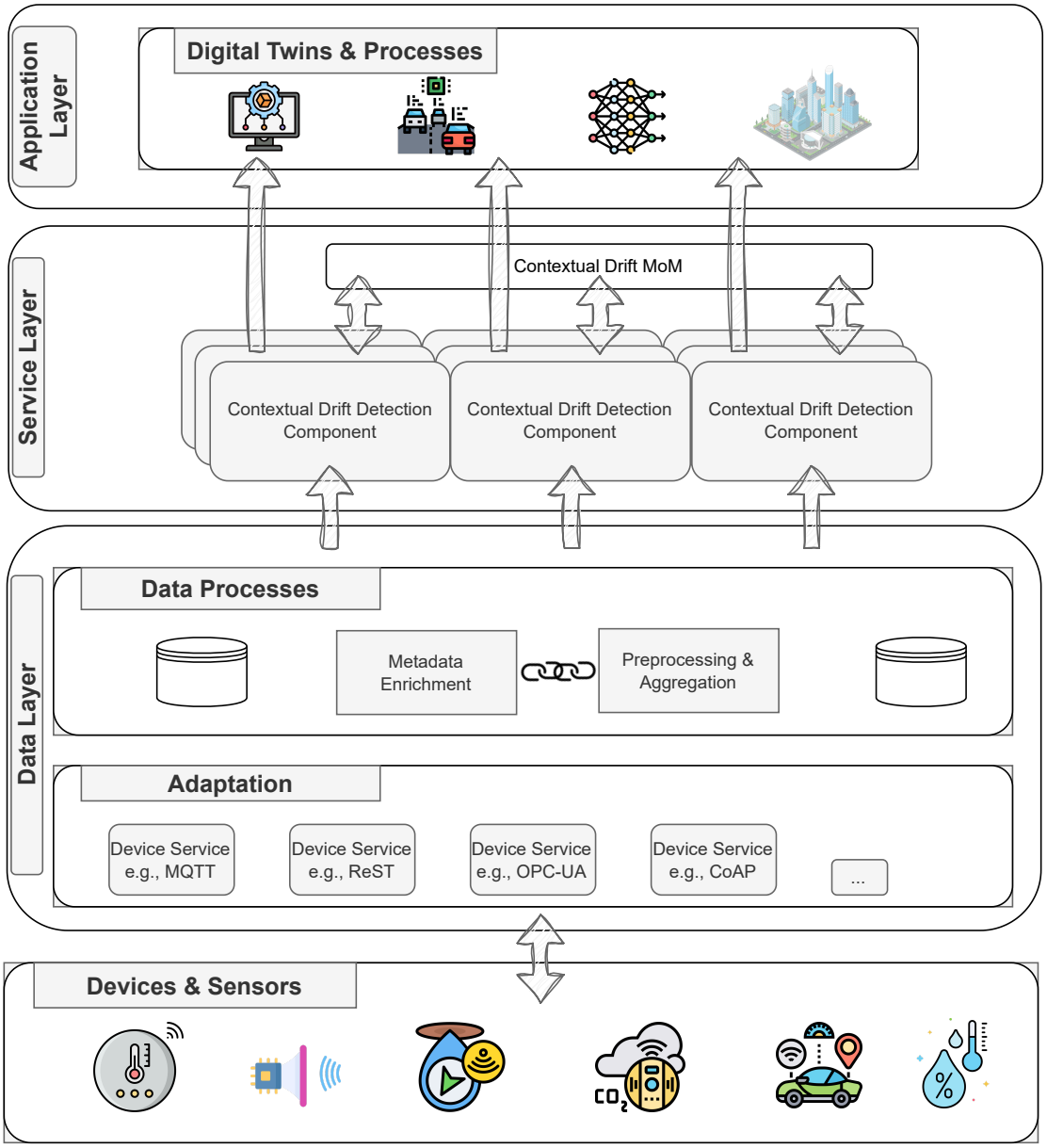


Figure 1: Multilayered architecture embedding context-aware Drift Detection.

- *Drift Classification*: when a drift is detected in the sensor data, the component broadcasts its drift status to nodes that subscribe to its channel. Then it enters a waiting period to allow neighbors to report their own drift events. The component then evaluates the drift reports from its neighbors to classify the nature of its own drift (Eq. 5). The local drift is classified as *systematic*, if the number of nodes in the component's neighbourhood experiencing a drift during the same time period exceeds a predefined threshold, indicating that a broader issue is affecting multiple nodes, such as an environmental or network-wide disturbance. Otherwise, it is considered *anomalous*, suggesting that it is isolated to the individual node or sensor.

The threshold for classifying a drift as *systematic* plays a crucial role in ensuring the robustness of the system. By requiring a fraction of neighbouring nodes to experience a drift concurrently, the

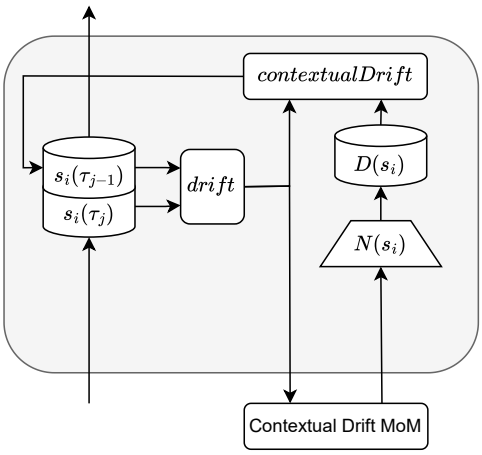


Figure 2: Context-aware drift detection component

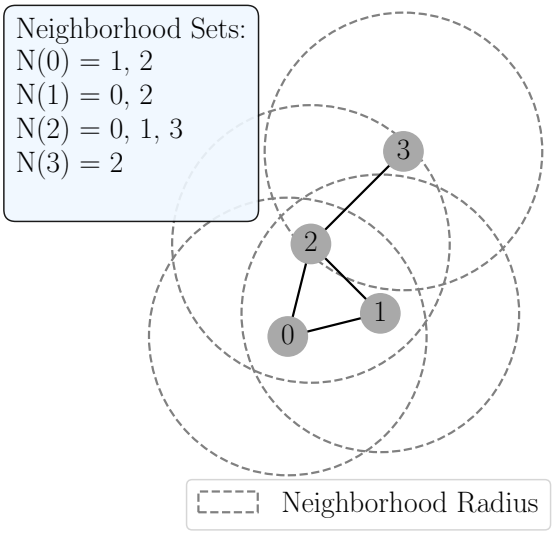


Figure 3: Representation of neighbourhood selection using a distance comparison.

system reduces the likelihood of misclassifications due to erroneous neighbour detection. This parameter can be adjusted to balance sensitivity and resilience based on the specific requirements and dynamics of the Smart City environment.

The data drift of each node is checked periodically. The period has to be selected so that the data gathered during that time is enough to reproduce the current data distribution. This is necessary to perform an accurate drift detection and also to have enough data to train the next version of the model. When an *anomalous* drift is detected the data is labelled as incorrect and it will not be used as a reference for future drift detections.

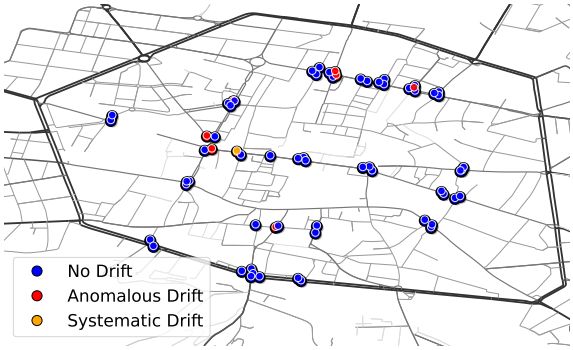


Figure 4: Map of context-aware drift detection in the first period. Sensors are localized by colored dots, blue indicates no drift, red anomalous drift and yellow systematic drift.

5 Evaluation

In our evaluation, we want to test the advantages of context-aware drift detection to identify *anomalous* drifts and retrain only for *systematic* drifts, consequently reducing the costs of model retraining during the management of model life cycles.

To evaluate our algorithm and framework, we tested it on data collected from the city of Bologna [1]. This dataset is formed by 66 different sensors distributed in the city center recorded in the year of 2024. Each sensor counts the number of cars that cross its sensed area each hour via a different process, and the same is done with the corresponding components.

The communication between components in the system is managed by EMQX [3], an MQTT messaging platform. MQTT is an appropriate selection for its capability of being used in any hardware, even low power sensors. Each sensor creates its topic on the EMQX broker to share its data. The same system is used for communication between components to find neighbors and share drift messages.

The *radius* for neighbour selection (Eq. 6) has been selected at 150 meters, being the mean street length in Bologna, ensuring that neighbouring sensors would have correlated car counts. The threshold h for contextual drift detection (Eq. 5) has been set to 0.5, as a standard majority vote.

Drift detection (Eq. 2) is performed using the two-sample Kolmogorov Smirnov test [13], a common algorithm for measuring the distance between two distributions. It is important to stress the fact that performing this measure has lower computational costs than training a deep learning model.

For the tests, we divided the year into three periods; the foremost is used as a reference, the first and the second are considered for drift detection. In Figure 4, 5 we represent the map of Bologna to show the types of drifts of each sensor using our framework. We selected these periods for their difference in drift, the first has few changes in distribution, while the second has a great *systematic* change given by prolonged city works. The difference in the number of *systematic* drifts is going to affect the ability of the system to detect *anomalous* drifts.

Following the tests performed with a year of data, we can find the different amounts of models trained by comparing the three

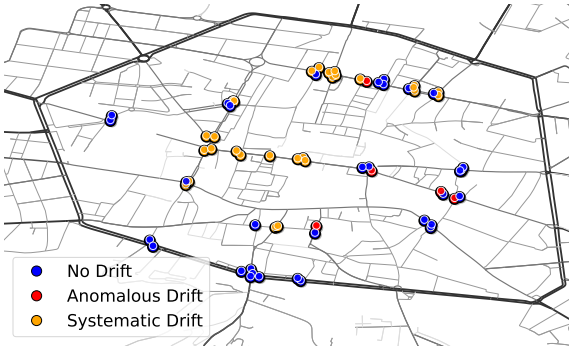


Figure 5: Map of context-aware drift detection in the second period. Sensors are localized by colored dots, blue indicates no drift, red anomalous drift and yellow systematic drift.

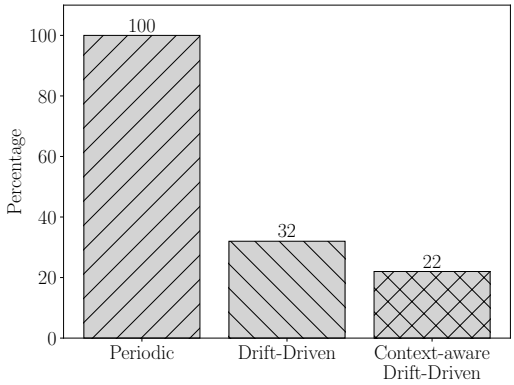


Figure 6: Relative costs with different retraining policies.

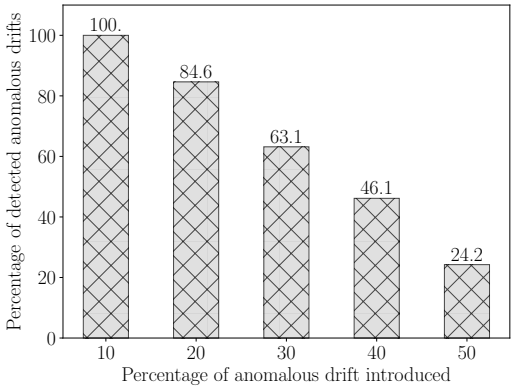


Figure 7: Percentage of detected anomalous drifts in the first period.

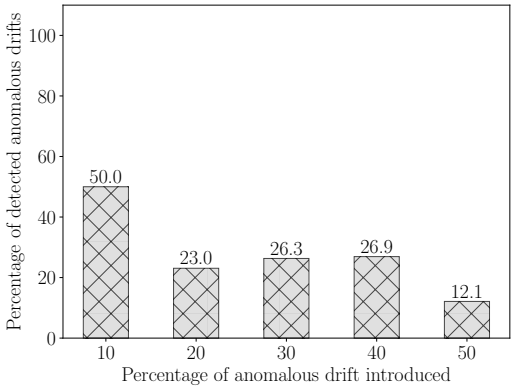


Figure 8: Percentage of detected anomalous drifts in the second period.

strategies for retraining: periodic, drift-driven and context-aware drift-driven (Figure 6). Using the Periodic retraining as a baseline reference, every model is retrained at each period. The Drift-driven approach cuts the number of trainings by 68 percent, avoiding retraining on non-drifted data sources. Using our context-aware drift-driven approach, we can further cut the costs by another 10 percent. This number can change based on the accuracy of the algorithm to detect *anomalous* drift and how many occur in the scenario. To further understand the algorithm behaviour in different situations, we conducted other controlled tests.

For each period, we introduced *anomalous* drifts in the dataset of an increasing amount of sensors, from 10 to 50 percent. In figure 7, 8 we can see the results in terms of percentage of detected *anomalous* drifts, increasing the number of attacks the context-aware drift detection efficiency deteriorates since it is based on neighborhood consensus, if the number of attackers reaches the number of good data distinguishing the two becomes progressively harder. Another weakness of the algorithm is the presence of *systematic* drift in the system, which helps *anomalous* drifts disguise themselves as *systematic* by being close to real drifts. This shows that improvements on the algorithm should be introduced to better discriminate these anomalies.

6 Conclusion

With the digitalization of SC, the costs of maintaining distributed intelligent systems increased with the advent of machine learning models and their high cost of training. We proposed a new way of reducing the number of model retrainings by introducing a new concept of *systematic* and *anomalous* drift. An algorithm to discriminate them, such as context-aware drift detection and an architecture to embed it in the smart city infrastructure. Through testing the real environment of the city of Bologna, we gave an idea of the possible reduction in costs of our proposed solution. In addition, we tested the accuracy of the algorithm in different scenarios, highlighting its strengths and weaknesses.

In the future, we expect the introduction of such discrimination algorithms in the smart city and their advancement in detecting *anomalous* drifts to increase city sustainability. We are currently

working on improving the detection algorithm, by introducing neighbouring selection based on data correlation between different kind of sensors. Instead of fixed radii and thresholds exploring adaptive methods that adjust based on sensor density, data type, or historical drift patterns. Moreover, we will conduct simulations or analysis focused on the communication and computational overhead introduced that can become a liability under large-scale deployment scenarios.

Acknowledgments

This work was supported by the Italian PRIN Digit4Circle (CUP: P2022788KK) funded under the Italian National Recovery and Resilience Plan (NRRP).

References

[1] [n. d.]. OPENDATA, comune di bologna. https://opendata.comune.bologna.it/explore/dataset/rilevazione-flusso-veicoli-tramite-spire-anno-2024, Accessed: 07/04/2025.

[2] [n. d.]. UNECE. https://unece.org/housing/smart-sustainable-cities, Accessed: 11-11-2024.

[3] [n. d.]. The Unified MQTT and AI Platform. https://www.emqx.com/en, Accessed: 05/06/2025.

[4] Faima Abbasi, Pierre Brimont, Cedric Pruski, and Jean-Sébastien Sottet. 2024. Understanding Semantic Drift in Model Driven Digital Twins. In *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems* (Linz, Austria) *(MODELS Companion '24)*. Association for Computing Machinery, New York, NY, USA, 419–430. doi:10.1145/3652620.3688256

[5] Paul Almasan, Miquel Ferriol-Galmés, Jordi Paillisse, José Suárez-Varela, Diego Perino, Diego López, Antonio Agustin Pastor Perales, Paul Harvey, Laurent Ciavaglia, Leon Wong, Vishnu Ram, Shihan Xiao, Xiang Shi, Xiangle Cheng, Albert Cabellos-Aparicio, and Pere Barlet-Ros. 2022. Digital Twin Network: Opportunities and Challenges. arXiv:2201.01144 [cs.NI] https://arxiv.org/abs/2201.01144

[6] Piotr Cal and Michał Woźniak. 2012. Drift Detection and Model Selection Algorithms: Concept and Experimental Evaluation. In *Hybrid Artificial Intelligent Systems*, Emilio Corchado, Václav Snášel, Ajith Abraham, Michał Woźniak, Manuel Graña, and Sung-Bae Cho (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 558–568.

[7] Qi Chen, Wei Wang, Fangyu Wu, Suparna De, Ruili Wang, Bailing Zhang, and Xin Huang. 2019. A Survey on an Emerging Area: Deep Learning for Smart City Data. *IEEE Transactions on Emerging Topics in Computational Intelligence* 3, 5 (2019), 392–410. doi:10.1109/TETCI.2019.2907718

[8] Oliver Cobb and Arnaud Van Looveren. 2022. Context-Aware Drift Detection. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.). PMLR, 4087–4111. https://proceedings.mlr.press/v162/cobb22a.html

[9] Hossein Darvishi, Domenico Ciuonzo, Eivind Rosón Eide, and Pierluigi Salvo Rossi. 2021. Sensor-Fault Detection, Isolation and Accommodation for Digital Twins via Modular Data-Driven Architecture. *IEEE Sensors Journal* 21, 4 (2021), 4827–4838. doi:10.1109/JSEN.2020.3029459

[10] Tianhu Deng, Keren Zhang, and Zuo-Jun (Max) Shen. 2021. A systematic review of a digital twin city: A new pattern of urban governance toward smart cities. *Journal of Management Science and Engineering* 6, 2 (2021), 125–134. doi:10.1016/j.jmse.2021.03.003

[11] Milan Groshev, Carlos Guimarães, Jorge Martín-Pérez, and Antonio de la Oliva. 2021. Toward Intelligent Cyber-Physical Systems: Digital Twin Meets Artificial Intelligence. *IEEE Communications Magazine* 59, 8 (2021), 14–20. doi:10.1109/MCOM.001.2001237

[12] Safwana Haque, Fadi El-Moussa, Nikos Komninos, and Rajarajan Muttukrishnan. 2023. A Systematic Review of Data-Driven Attack Detection Trends in IoT. *Sensors* 23, 16 (2023). doi:10.3390/s23167191

[13] J. L. Hodges. 1958. The significance probability of the smirnov two-sample test. *Arkiv för Matematik* 3, 5 (01 Jan 1958), 469–486. doi:10.1007/BF02589501

[14] Hisashi Kanda, Hiroyuki Okamura, and Tadashi Dohi. 2023. A Note on Optimal Retraining Strategy for ML Systems. In *2023 10th International Conference on Dependable Systems and Their Applications (DSA)*. 730–733. doi:10.1109/DSA59317.2023.00104

[15] Ansam Khraisat and Ammar Alazab. 2021. A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges. *Cybersecurity* 4, 1 (08 Mar 2021), 18. doi:10.1186/s42400-021-00077-7

[16] Michail Loufakis, Othonas Manis, Christos Kioroglou, Nikolaos Kolokas, Dimosthenis Ioannidis, Dimitrios Tzovaras, and Mile Stankovski. 2024. Intelligent Model Management: Using Reinforcement Learning to Detect Data Drift and Retrain Industrial Machine Learning Systems. In *2024 26th International Conference on Digital Signal Processing and its Applications (DSPA)*. 1–6. doi:10.1109/DSPA60853.2024.10510082

[17] Ananth Mahadevan and Michael Mathioudakis. 2024. Cost-aware retraining for machine learning. *Knowledge-Based Systems* 293 (2024), 111610. doi:10.1016/j.knosys.2024.111610

[18] Hassan Mehmood, Ahmed Khalid, Panos Kostakos, Ekaterina Gilman, and Susanna Pirttikangas. 2024. A novel Edge architecture and solution for detecting concept drift in smart environments. *Future Generation Computer Systems* 150 (2024), 127–143. doi:10.1016/j.future.2023.08.023

[19] Hassan Mehmood, Panos Kostakos, Marta Cortes, Theodoros Anagnostopoulos, Susanna Pirttikangas, and Ekaterina Gilman. 2021. Concept Drift Adaptation Techniques in Distributed Environment for Real-World Data Streams. *Smart Cities* 4, 1 (2021), 349–371. doi:10.3390/smartcities4010021

[20] Qingfei Min, Yangguang Lu, Zhiyong Liu, Chao Su, and Bo Wang. 2019. Machine Learning based Digital Twin Framework for Production Optimization in Petrochemical Industry. *International Journal of Information Management* 49 (2019), 502–519. doi:10.1016/j.ijinfomgt.2019.05.020

[21] M. Mazhar Rathore, Syed Attique Shah, Dhirendra Shukla, Elmahdi Bentafat, and Spiridon Bakiras. 2021. The Role of AI, Machine Learning, and Big Data in Digital Twinning: A Systematic Literature Review, Challenges, and Opportunities. *IEEE Access* 9 (2021), 32030–32052. doi:10.1109/ACCESS.2021.3060863

[22] Daniel Sigalov and Nahum Shimkin. 2011. Cross Entropy Algorithms for Data Association in Multi-Target Tracking. *IEEE Trans. Aerospace Electron. Systems* 47, 2 (2011), 1166–1185. doi:10.1109/TAES.2011.5751250

[23] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. 2014. Internet of Things for Smart Cities. *IEEE Internet of Things Journal* 1, 1 (2014), 22–32. doi:10.1109/JIOT.2014.2306328