Future Generation Computer Systems 150 (2024) 127–143

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

A novel Edge architecture and solution for detecting concept drift in smart environments

Hassan Mehmood [a,*], Ahmed Khalid [b], Panos Kostakos [a], Ekaterina Gilman [a], Susanna Pirttikangas [a]

[a] University of Oulu, Pentti Kaiteran katu 1, 90570 Oulu, Finland
[b] DELL Technologies, Cork, Ireland

ARTICLE INFO

Article history:
Received 15 November 2022
Received in revised form 21 August 2023
Accepted 23 August 2023
Available online 26 August 2023

Keywords:
Edge computing
Cloud–edge continuum
Data centres
Concept drift
Smart cities
Smart buildings
Decision-making
Big data
Machine learning
Data analysis

ABSTRACT

The proliferation of the Internet of Things (IoT), artificial intelligence (AI), the adoption of 5G, and progress towards 6G technology have led to the accumulation of massive amounts of real-world data; however, a significant portion of the data generated by smart cities and smart buildings remains unused. A notable problem is the shift of statistical properties in real-world streaming over time caused by unexpected factors, referred to as concept drift, which results in less efficient predictive models. To address this problem, the latest research leverages the cloud–edge continuum paradigm for the deployment of AI and general smart city applications while utilising the available resources optimally. In this article, we propose a computing architecture for different smart city applications in edge micro data centre (EMDC) settings over a hybrid cloud–edge continuum to support the deployment of AI workloads. We implement a feedback-driven automated concept drift detection and adaptation methodology, combining base learner long short-term memory (LSTM) with Page–Hinkley test (PHT), adaptive windowing (ADWIN) and the Kolmogorov–Smirnov windowing (KSWIN). Real-world data streams are utilised to forecast from various environmental sensors installed at the University of Oulu Smart Campus. The feedback-based concept drift detection and adaption process is first evaluated using synthetic datasets with known concept drift points and then employed in the real-world data. Subsequently, the implementation is evaluated using the state-of-the-art MAE, RMSE, and MAPE methods. The results showed a reduction in MAPE from 8.5% to 3.88% when concept drift detection was applied. Additionally, the challenges faced and the effectiveness of the suggested solutions are explored.

1. Introduction

The Smart City of the post-cloud era is often envisioned as a world in which virtualised resources deliver on-demand computing power and services [1]. These resources are managed through multiple layers of abstraction and distributed across geographically sparse and decentralised networks [2]. Enabled by innovations in edge computing and the hybrid cloud–edge computing continuum, the post-cloud era is said to improve technology integration drastically, providing connectivity and *in situ* intelligence even to remote urban and rural areas. However, this vision overlooks one key aspect of the city: its physical infrastructure and polycentric fabric are not simply static structures [3]. The cities of the future, which are predicted to host more than 68% of the world's population by 2050, face many disruptive natural and man-made challenges [4] that have led to re-localisation. As recently observed with the outbreak of the global Covid-19 pandemic, these challenges can abruptly transform how public and private spaces are used and can therefore trigger socioeconomic behaviours that can weaken decision-making models [5].

Today, smart cities worldwide are already automating processes in public spaces, using sensors to monitor traffic lights and water levels, manage streetlights and public transport schedules, and collect data from various sources to improve the efficiency of city services. Residential smart and connected spaces have also seen a surge in demand, with the smart building market projected to increase from about USD 67 billion in 2021 to USD 328 billion by 2029 [6], while the smart homes market is anticipated to reach USD 262.63 billion by 2025 [7]. Smart spaces infer intelligence insights by leveraging a plethora of ubiquitous data-generating devices, "things", and applications that are becoming strongly embedded in our everyday life and routines [8]. Advances in key

* Corresponding author.
E-mail addresses: hassan.mehmood@oulu.fi (H. Mehmood), ahmed.khalid@dell.com (A. Khalid), panos.kostakos@oulu.fi (P. Kostakos), ekaterina.gilman@oulu.fi (E. Gilman), susanna.Pirttikangas@oulu.fi (S. Pirttikangas).

https://doi.org/10.1016/j.future.2023.08.023

enabling technologies (KETs) such as AI, edge intelligence, and 5G networks are the catalysts for delivering the vision of the smart city in the post-cloud era where data production takes place at the far edge of the network near the user, and the processing can be distributed across a range of diverse computing units.

Nevertheless, cities are complex systems composed of multiple physical elements (e.g. parks, streets, buildings, and other spaces/places). These elements are not simply static structures whose functionality remains unchanged over time. While the semiotics of the phrase "edge of the network" incorporates the notion of uniformity, the edge is *stricto sensu* more fluid than the core.

Smart city ecosystems are built using a combination of different technological alternatives such as sensors, IoT, cloud & edge computing, cyber–physical systems, and other technologies, producing an overwhelming amount of data [9–12]. Moreover, the availability of large-scale data brings its own challenges, such as data management, computational demands, data quality, integration with machine learning methods for insights, ensuring the ground truth, and others [9,13].

Previous studies have highlighted the need for edge computing to address critical geographical dependencies entailed in by the fluidity of spaces/places whose utility may change over time at a different rate [14]. The emerging consensus is that the edge is not an environment denoted by stationary processes in the sense that historical patterns and behaviours are not necessarily expected always to be repeated in the future [13,15]. This assumption is supported by the fact that data produced and consumed at the edge of the network and the machine learning (ML) models delivering edge intelligence are susceptible to concept drift that can cause severe model divergence and ineffective decisions.

In many smart city application domains, when forecasting, the data can eventually become obsolete due to statistical variations or changes in the context of the data [16]. For example, forecasting environment measurements such as the CO_2 abundance, air quality, or humidity from sensors can become outdated due to differences in calibration and malfunctioning devices. Such a shift is known as concept drift, and its prompt detection is critical for services to run at the required quality. Although concept drift can be seen in many real-world scenarios, few research results are available in which learners integrated with concept drift detection methods for time series data are used. [17]. In addition, most of the literature on concept drift focuses on classification tasks, whereas regression problems still require exploration [18]. Moreover, use cases resembling smart city applications with large datasets rarely incorporate concept drift detection and adaptation, which requires more exploration and experimentation [13].

To address the concept drift in data, different frameworks have been developed considering both computational and data-related challenges. Technology is advancing rapidly in this context, and various cloud computing-based solutions have been proposed to tackle this challenge [13,19–21]. While data variability has long been considered a critical problem in big data analytics, there have been previous efforts to better understand the problem of concept drift at the network's edge [22–24]. The cloud computing paradigm has significantly helped in meeting the high computational needs of smart cities. However, cloud computing presents difficulties of high latency and a substantial network overhead for data collected by smart city sensors or IoT deployments [25]. In contrast, edge computing promises lower latency than cloud computing, less computational overhead, more scalability and fault-tolerance, exhibiting better characteristics [23–25]. As a result, there is a clear need for more adaptable hybrid cloud–edge computing-based smart city systems to produce more robust and accurate municipal services.

This paper proposes an edge architecture in EMDC settings over a hybrid device-edge–cloud continuum. The solution supports the development of data-driven intelligent smart city services, including data collection, processing and analysis. Moreover, EMDC addresses the challenges of automated concept drift handling at the edge, a gap we have found in related work. To demonstrate the feasibility of our solution, we have implemented automated concept drift detection at the edge of the sensors of a smart building (Smart Campus, University of Oulu) in an emerging smart city.

Therefore, the main contributions of this paper are:

1. A novel containerised architecture to handle computationally diverse data pipelines based on an edge computing paradigm.
2. An innovative approach to automated concept drift detection using a feedback-based algorithm from the real-world sensor data streams of a smart building.
3. To support the integration of different smart city scenarios and workflows for decision-making.
4. To facilitate on-demand live migration of workloads (e.g., model training, data resampling, etc.) across the hybrid cloud–edge continuum based on available resources.

The remaining sections of the paper are organised as follows: Section 2 presents a literature review on advances in smart cities, a review of smart city platforms, and concept drift detection methods. Section 3 presents a high-level view of edge computing architectures with a newly proposed solution. The proposed EMDC implementation is verified with a use case described in Section 4 and experiments in Section 4.2. Section 5 gives the results of the performed experiments. Later, Section 6 discusses the learning outcomes, challenges, and limitations of the work. Finally, Section 7 concludes the paper.

2. Literature review

It is commonly recognised that smart city data is heterogeneous and originates from diverse sources, requiring robust data management techniques for exploration and decision-making [9]. This section describes the current data management challenges and how they are addressed using state-of-the-art tools and frameworks. We also present a brief literature review on concept drift detection approaches.

2.1. Recent advances in smart cities

Recent decades have seen the development of technologies that are making cities and buildings smarter [26]. According to current projections, the world's population will grow by 30% by 2030, 60% of people will live in cities, and 43 megacities will be home to more than 10 million people [27]. Indoor spaces, where people spend an average of 90% of their time [28], are crucial to the smart city ecosystem. Consequently, the idea of a smart campus serves as a natural progression from the smart city paradigm, emerging from nearly three decades of collaborative work between researchers, educators, and technology professionals [29,30]. The objective is to enhance efficiency, sustainability, and engagement for all campus community members, regardless of their physical presence or online participation. Over the past decades, these efforts have focused on incorporating cutting-edge and developing technologies into educational environments, such as indoor positioning, navigation, occupancy detection, and environmental monitoring. Specifically, a recent systematic survey investigated 44 systems related to smart campuses, encompassing a diverse array of physical infrastructure elements, including smart transportation, autonomous vehicles,

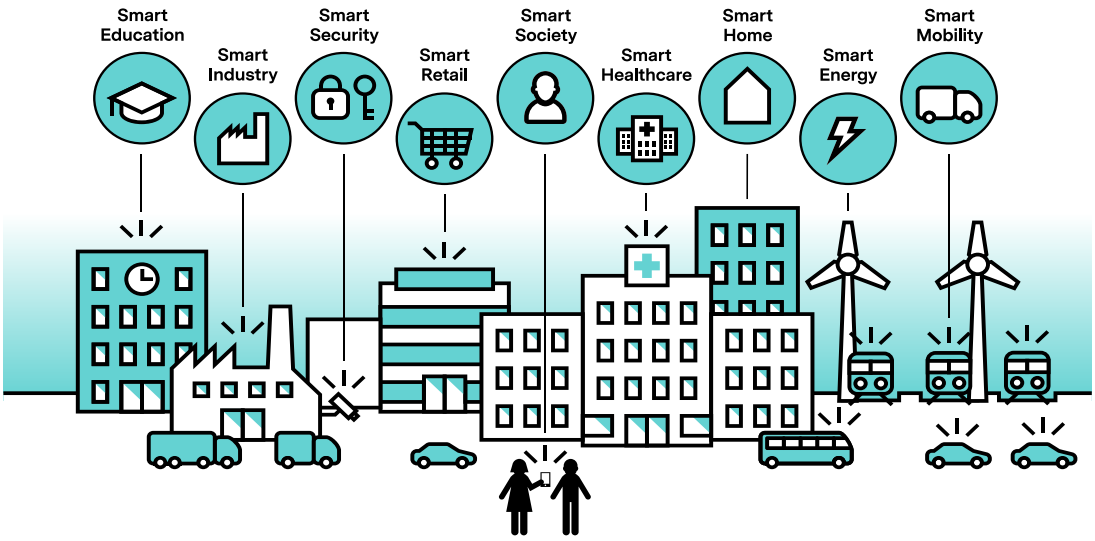


Fig. 1. Different dimensions in smart cities.
Source: Adapted from [32].

buildings, smart classrooms, laboratories, public spaces, parking, and lighting. Smart transportation can be further divided into smart traffic lights and electric vehicles, while autonomous vehicles can be characterised by Unmanned Aerial Vehicles (UAVs) and Connected and Autonomous Vehicles (CAVs) [31].

Evidently, the services developed based on the IoT, and the multitude of sensors with cost-effective computational resources have made the ever-growing urban areas and buildings intelligent. These developments are now evident in public offices, smart buildings, utility applications, the transportation domain, earth-moving machines in construction and mining areas, health care, economic platforms, and others [9,26,33,34]. Furthermore, such technological disruption not only makes cities and buildings smarter but also helps to improve people's well-being and living standards via smart apps such as healthcare portals, e-learning platforms, routine planning applications, weather applications, and others [13,33]. The solutions and services created for smart cities are projected to increase by more than USD two trillion worldwide [35]. While the concept of smart cities remains elusive, modern cities rely increasingly on emerging technologies such as 5G, the IoT, distributed cloud infrastructures, edge computing, smartphones, and artificial intelligence to help optimise governance and management solutions and enact new policies using the available data. [9,33].

A typical smart city consists of several dimensions, each equally playing a pivotal role towards a smarter society. Fig. 1 shows nine dimensions that form a generic smart city ecosystem: smart industry; smart security; smart energy; smart people; smart health; smart retail; smart governance; smart home; and smart mobility [32]. The bulk of these innovation efforts have the support of governments, which want to make cities accessible and safe [31,36] for both new and seasoned city dwellers [9,37]. Such a rapid innovation ecosystem requires deep support from KETs such as the IoT, cloud computing, big data, edge computing, and cyber–physical systems. Smart cities' promise of digitalisation necessitates additional deployments of IoT or a multitude of the aforementioned technologies, posing enormous challenges [9, 26,37]. Examples of such challenges include ensuring the security and privacy of smart city platforms ([38], data heterogeneity issues, sensors and their connectivity, big data analytics, scalability, reliability, and the fault-tolerance of services [27,37]. There have been several surveys on these subjects [33,37,39].

2.2. Smart city initiatives

When combined, the above technologies provide enormous prospects of creating various applications to run cities better and establish new urban management rules by using insights from smart city data. Currently, many smart city initiatives are harnessing data from diverse sources within smart cities for the betterment of society [13,39]. Despite the above hurdles, numerous smart city initiatives have attempted to include various components (e.g., smart mobility, smart transportation, smart energy, etc.) either separately or collectively. For example, a study has surveyed enabling techniques to reduce waste, pollution hazards, energy consumption, and reducing traffic towards making a city safer, and more sustainable and eco-friendly [39]. Tokyo has announced plans to become the most environmentally friendly city in Asia by 2020, with objectives for effective transportation management, reduced energy use, and other areas. [40]. Most municipal solid waste in Tokyo is now repurposed, dwellings are built to be fire and earthquake resistant, and the creation of parks and communal paths as part of disaster management is underway.[41]. Similarly, next-generation initiatives like U-city are committed to bringing cutting-edge infrastructure and ubiquitous services to 15 Korean cities. Part of the initiative, Hwaseong-Dong tan U-city was completed with a focus on crime prevention, intelligent traffic flows, and the automation of urban space infrastructure [42]. With its explosive population growth, China aimed to invest billions towards approximately 500 smart cities to incorporate smart services, improved infrastructure, AI and big data, and digitised regulation management by 2020 [43–45]. Similarly, the USA finalised a smart city plan intending to invest about USD 41 trillion by 2035 to create an IoT-based smart city infrastructure to support improved air quality, efficient transportation, sustainable environment, and other initiatives [46].

In Europe, under the European Innovation Partnership on Smart Cities and Communities (EIP-SCC) initiative, about 300 smart city projects are supported with up to EUR 1 billion to tackle the challenges of energy-efficient buildings, smart mobility, digital infrastructure, improved living standards, sustainable climate, and others [47]. In a study by [48], an IoT infrastructure was deployed to enable the process of environmental monitoring, transportation management, surveillance, and others. A smart city platform was developed under an EU-funded

Table 1
Overview of smart city architectures.

Project name	Goal	Technologies
SmartSantander [48]	An IoT testbed was proposed to integrate a heterogeneous set of devices (actuators, NFC, sensors, etc.) using different network technologies to enable monitoring and smart city services and applications.	A three-tiered server architecture was implemented with server tier, IoT gateway tier, and IoT device tier.
Sii-Mobility Smart City [53]	A project platform which aims to provide cutting-edge solutions for sustainable mobility and transportation using cloud-based solutions.	A multi-tier architecture was proposed to support big data analytics using tools such as Hbase, HDFS, Apache Zeppelin and others.
CiDAP [54]	To address the challenges of data storage and processing from SmartSantander [48], a big data platform was introduced supporting both historical and real-time processing.	The cloud-based CiDAP platform was developed using HDFS, Apache Spark, CouchDB etc.
Snap4City [55,56]	A platform which provides flexible opportunities to create smart city applications using heterogeneous data enabled by IoT, data analytics, and big data technologies.	A cloud infrastructure was implemented using a combination of different toolsets, e.g. MQTT, NodeRED, Hbase, Apache Nifi, Apache Spark etc.
The CUTLER Project [9,49]	A platform was proposed to collect the data from existing sensing infrastructure within smart cities and shift the intuition-based decision-making process to an evidence-based one through data-driven insights.	The implemented hybrid cloud distributed infrastructure comprised several technological solutions such as Apache Hadoop, Apache Kafka, ELK, and Camunda.
The iSapiens platform for smart city applications [57]	A multi-layered architecture was introduced for smart city applications supporting online analytics. However, tasks with demanding resources are executed on off-network cloud services as offline analyses.	A distributed edge computing paradigm-based platform was introduced providing connectivity for devices, analytics, and an off-network DBMS and web interface.
Framework for an IoT based smart city [2]	A layered approach was presented for situation awareness in IoT-based smart cities using an edge computing paradigm. However, the controlling and data processing are done using cloud-based middle-ware through distributed edge servers.	The system uses several technological solutions for implementation, such as WSN, MQTT, Cassandra, and other supported tools.
The ATCLL platform [58]	Aviero Tech City Living Lab developed a data platform to support data collection, processing, and visualisation focusing on analysing environmental data, mobility patterns, and network feeds. It also enables the integration of third-party services for exploration, application development, and experimentation.	The platform utilised open-source frameworks and the FIWARE environment with integrated tools such as Apache Kafka and MongoDB.
The CTwin platform [59]	A platform that provides real-time situational awareness related to urban transportation through cyber–physical controls, analytics dashboards, and simulations.	A cloud computing paradigm-based distributed platform was developed using Docker, Kubernetes, Angular, and others.
Smart Geo Layers (SGeoL) [60]	SGeoL was built for developing smart city applications and integrating urban data. The platform supports various abstractions, such as data management, data storage and analysis, visualisation, and others.	A combination of open source frameworks such as Apache Flink, HDFS, Vue.js, and others are used to develop the platform on cloud infrastructure.

project called CUTLER to provide evidence-based policy-making focusing on coastal urban development by utilising big data from five different city pilots (Thessaloniki, Antalya, Vicenza, Cork, Antwerp) [49]. The project involved multiple use cases, equally contributing towards better urban management policies, e.g., controlled parking systems and flood risk reduction [9,50,51].

Finally, smart campuses, historically known as catalysts for nurturing emerging technologies, have undergone substantial changes in the education market in recent years. These shifts have been propelled by the dual forces of the green transition and the Covid pandemic, which prompted the adoption of the IoT and cloud-based technologies such as Unified Communications as a Service (UCaaS), Communications Platform as a Service (CPaaS), and Contact Centre as a Service (CCaaS) [52].

2.3. Review of smart city platforms

The concept of smart cities was introduced to improve the living standards of citizens and provide better public administration through ICT [61]. It is therefore important for smart cities to take precautions to protect the public assets that may be at risk. This includes a wide variety of systems that have been used for a long time (e.g. legacy systems) but have not been updated, such as those used by the government, universities, legal infrastructures, power plants, the transport system, healthcare systems, community platforms, and environmental sensing systems. [53,62].

A great deal of work has gone into designing application-specific components of smart cities, such as parking management and waste disposal. However, there is much less research on integrating such solutions into a holistic smart city platform [24,32].

Various issues are involved in the realisation of smart city ecosystems, especially the management and processing of the vast amount of data [9]—for example, data collection, management, storage and processing challenges, analysis and visualisation. Embedding big data technologies and a distributed computing paradigm in smart city platforms are a perfect fit for such cases. Efficient and innovative smart city platforms are still in demand due to deteriorating progress focusing on individual aspects such as environment sensing, energy management, and others. Furthermore, the rapid data growth of smart cities also reduces the performance of complex data processing platforms. For example, cloud computing has reshaped traditional computing, and now with 5G adoption, robust edge computing paradigm-based data processing platforms are needed in the context of smart cities. In Table 1 below, we provide a review of existing smart city solutions developed to address these challenges along with their goals and shortcomings.

Although several smart city platforms have been put in place, none are yet complete. [57]. In addition, with the progression of the edge computing paradigm, there is a need for hybrid cloud and edge-based smart infrastructures that can support distributed

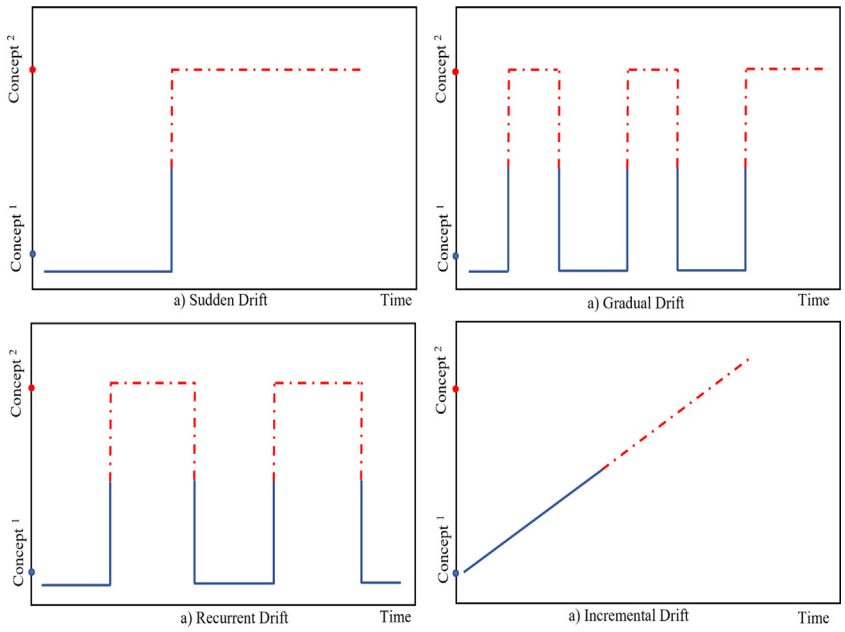


Fig. 2. Different types of concept drift.

data processing and analysis. The idea of edge computing differs from cloud computing, where data is moved to a centralised entity for analysis. In edge computing, the computing occurs near the data source providing energy efficiency and low-latency benefits [2]. As Table 1 shows, the majority of the proposed solutions are based on the cloud, creating the demand for more edge computing-based smart cities architectures to tackle the challenges of low latency, providing intelligence at the edge, less movement of data to the cloud, and the provision of computational resources at the edge. In addition, to the best of our knowledge, the proposed edge computing-based architectures are incapable of resource-demanding analysis at the edge of the network and do not support the live migration of workloads, considering efficient resource utilisation, See Table 1.

In this regard, the concept of EMDC could be utilised for future smart infrastructures in smart cities that are equipped with enough resources and technological solutions. In this study, we propose an EMDC-based architecture (Section 3) for the development of smart city applications that creates a continuum of far-edge and near-edge resources for handling and processing time-sensitive applications and offloading less-critical tasks to the cloud.

2.4. Concept drift detection and adaptation

Concept drift is a well-known challenge in streaming data when performing predictive modelling. The phenomenon occurs because of the dynamic behaviour of streaming data, i.e. sudden or gradual changes in the statistical properties of data for unanticipated reasons. For example, a concept drift can occur in data due to a malfunctioning or differently calibrated sensor device or a change in the data context [13,63]. There are different types of concept drift, such as sudden, gradual, and recurring concept drift [64], see Fig. 2. In sudden drift scenarios the drift can happen abruptly, while in gradual or incremental drifts, changes in data occur slowly with an additional change of class distribution in gradual drift, whereas in recurring drift the change in the data can vanish and return after a certain period [64,65].

The current state-of-the-art provides numerous concept drift detection and adaptation methods based on active and passive approaches. In active approaches, explicit concept drift detection is required before employing an adaptation approach to the learner. Meanwhile, in passive approaches, the learner can continuously update without needing to know what kind of concept drift has occurred and when. However, finding the optimal frequency of learner updates can require a lot of effort [13].

There is substantial literature on both active and passive approaches for concept drift detection and adaptation. The active approaches are usually categorised into three main categories: (i) sequential analysis-based methods (ii) window-based methods (iii) learner output-based methods [13,66]. Passive approach methods consist mainly of single classifiers and ensemble-based learners [67]. This section provides a short survey on each of these approaches from the existing literature. For more details, please refer to [66,68–70].

Sequential analysis-based methods sequentially examine the data stream to analyse the change. When a defined threshold is exceeded via change detection from the data distribution, an alarm is generated that a concept drift has been detected [66]. Examples of sequential analysis-based algorithms include the Page Hinkley Tests (PHT) and the Cumulative Sum (CUSUM) [66]. Specifically, PHT has been used as a benchmark in the previous literature [13,71,72]. PHT works by computing the difference in the observed values and their mean, if the observed mean is greater than the defined threshold, a concept drift has been detected [73].

Learner output-based methods operate by tracking the error rate of the learner's output. Predicted values are analysed to detect concept drift in streaming data by monitoring their standard deviation and the mean [13,66]. Some examples include the Drift Detection Method (DDM) [68], the Early Drift Detection Method [74], the Reactive Drift Detection Method (RDDM) [75], and EWMA for Concept Drift Detection (ECDD), which utilises EWMA charts for concept drift detection [70,76].

Window-based methods detect concept drift by monitoring the change in the distribution of predicted values. Two subsets of windows are employed in such methods, where one window contains the original data distribution and a sliding window that holds the predicted values to indicate the statistical differences [13,66]. Methods such as Adapting Windowing (ADWIN), Drift Detection Method based on Hoeffding's inequality

(HDDM$_{A-Test}$) [70], HDDM$_{W-Test}$), and the Fast Hoeffding Drift Detection Method (FHDDM) [77] belong to this group.

Apart from the previously mentioned approaches to concept drift detection, passive approaches are also used, largely to deal with concept drift in non-stationary environments. However, passive approaches do not specifically require the explicit detection of concept drift. Instead, the learners are retrained continuously based on predefined frequencies [13,65].

3. Edge Micro Data Centre (EMDC) architecture

Table 1 lists a variety of recent smart city architectures. Most of these solutions build a smart city platform using virtual machines (VMs), e.g., [49,58,60]. While this is more flexible than physical machines when establishing cloud-based infrastructures, VMs are restrictive when moving user or system applications to different locations. In addition, some architectures carry out resource-demanding processing either off-network or through cloud middleware [2,57]. A recent study by [59], provides a containerised generalised smart city architecture for different scenario-driven smart city applications. Despite the holistic nature of the architecture, it shows limited capabilities for system-level logs and lacks telemetry components, which are essential for determining resource requirements for the workloads in the long run.

We aim to build a portable architecture that allows us to integrate different smart city scenarios easily, and that supports concept drift detection, facilitates continuous monitoring of the logs, and enables on-demand live migration of workloads (AI, ML, and general workloads) across a cloud–edge continuum based on the resource requirements. Specifically, the novelty of our architecture can be summarised as follows:

- Support for computationally demanding analysis (data processing, predictive modelling, concept drift detection, etc.) in EMDC settings.
- Facilitation of easy integration of different smart city scenarios and workflows for decision-making.
- Support for on-demand live migration of workloads across clusters and computing nodes based on resource requirements.
- The implementation of monitoring and telemetry tools to continuously monitor resource usage and enable analytics-driven resource requirements of workloads migration.

3.1. Hybrid computing architecture

To build an agile architecture and resource organisation that enables rapid access to actionable intelligence and big data processing, we define a hybrid device-edge–cloud continuum. As shown in Fig. 3, rather than examining the edge and cloud as two separate computing paradigms, we create a powerful continuum of computing, storage, and network resources, that starts at the source of data generation and ends with extremely high-performance resource hubs in the cloud. The AI/ML workloads are scheduled to run in different environments across the continuum based on the type, properties, and requirements of the workload. In the remainder of this section, we provide a brief overview of various entities in our infrastructure, including user devices, edge servers and cloud computing.

Devices: In general, we consider devices to be the on-premises entities that generate data, e.g., sensors that measure environmental parameters, actuators, cameras that record video streams, smart vehicles, or traffic signals, and motion detectors. Modern devices may also be embedded with minimal computing or storage resources and capabilities of initial and minimal data

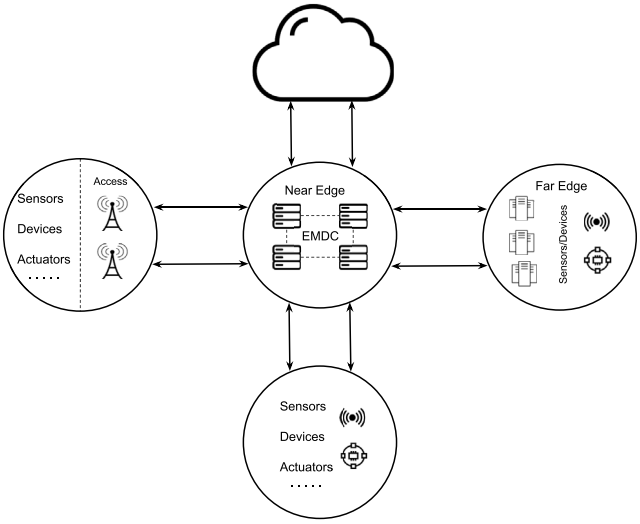


Fig. 3. Hybrid device-edge–cloud continuum for smart infrastructures and cities.

processing [78], e.g., for filtering data, aggregating measurements, and discarding invalid entries.

Edge: The edge encompasses a broad array of execution venues but is generally divided into the near edge and far edge, based on the amount or performance of computing resources available. The near edge either resembles a traditional data centre located closer to an end-user but with lesser compute and storage capacity or it consists of powerful Edge Micro Data Centres (EMDCs) with multiple computing and storage slots equipped with GPUs and hardware accelerators [79]. User devices may connect with the near edge through an access network, such as 5G. The far edge is generally located on-premises or very close to the user devices and is further constrained in terms of resources. The far edge may consist of traditional servers or resource-constrained EMDCs as standalone units that can be connected to multiple user devices.

Cloud: Cloud computing consists of the latest powerful generation of fast and efficient computing and storage hardware. By leveraging the capabilities of the distributed computing paradigm, cloud computing is being utilised to address many challenges where data-intensive solutions are needed. For example, cloud computing applications are present in big data analytics, healthcare, economic platforms, policy making, and others [13]. However, as mentioned in Section 2.3, cloud computing can add high latency and network overhead challenges with data becoming more real-time in nature, and in real-time decision-making systems where insights are generated in real-time. Advanced solutions are therefore needed based on edge computing or hybrid cloud–edge architectures, where such challenges can be addressed.

3.2. EMDC design

As the section above explains, our edge ecosystem consists of Edge Micro Data Centres (EMDCs) capable of processing data close to its source. To meet the requirements of scalability, diversity, and reliability, the EMDC is designed to run a containerised environment. Traditional environments use virtual machines to distribute physical resources among various components and services required to process the data. This approach results in a significant overhead needed to manage the virtual machines, and it reduces the flexibility in distributing resources between workloads, and slows down the spawning of the machines and processes. Compared to the traditional approach of virtual machines (VM), containers are more lightweight, portable, granular,

and easy to deploy and manage. As such, we design our architecture to run in a completely containerised environment, with all the AI workloads running as microservices on the EMDC.

The EMDC consists of multiple slots to support multiple storage and processing cards. We use Kubernetes (kubernetes) to orchestrate and manage the resources available at these slots. Each slot is configured with a Linux OS. One slot is configured as a kubernetes master node, also known as a control plane. This node acts as the central management node and handles all orchestration, scheduling, and management tasks. The remaining slots are configured as kubernetes worker nodes and are used to run the workloads. The scheduler is augmented with the results of the concept drift detection (Section 4) to determine the needs of an AI/ML workload. When a workload has to be moved from the edge to the cloud or more powerful edge servers, our architecture performs a real-time live migration and processes the more expensive operations in the cloud. Once the expensive retraining is complete, the workload is moved back to the edge, and the operation continues without interruption.

We use Docker for the run time and virtualisation of applications. Our AI/ML workloads are packaged into Docker images along with all the dependencies and uploaded to an image repository. These workloads are then scheduled as microservices by defining and deploying them as Kubernetes objects. Kubernetes handles the orchestration and placement of the workloads on one or more of the worker nodes based on each workload's requirements, specifications, and constraints. In contrast with the traditional VM approach, this deployment is more dynamic and does not require pre-planning and the static allocation of resources (such as CPU, memory, or storage). A workload will only consume the resources that it requires, and the remaining resources on that worker node can be dynamically allocated to other workloads as needed.

In Kubernetes, the workloads are scheduled as objects called pods. A pod generally consists of one container running a specific application. However, depending on the application design, multiple containers can be packaged into one pod if required. A pod is scheduled to run on one worker node. However, for fault tolerance, a replication factor greater than one can be used to create replicas of pods and schedule them on more than one node to avoid a single point of failure. We use a Kubernetes object called a Deployment to achieve this for our system components and applications that require a higher fault tolerance.

To enable communication between workloads, we use Kubernetes objects called Services. Kubernetes services expose an application running on a set of pods as a network service to other internal or external applications. These services abstract the network functionality and extract from the applications, making the applications dynamic and easier to manage. By default, containers are ephemeral and do not maintain any state or data from previous runs. For our applications that require persistent storage, we use Kubernetes Persistent Volumes (PV) and mount these PVs into the pods running our applications.

Finally, we use Kubernetes objects called ConfigMaps to store all the dynamic configuration and environment variables of our applications and pass them to the application at run-time. This decouples configuration from the main application and makes the application more portable. The design of containerised platform can be seen in Fig. 4.

3.3. Platform implementation

A smart city platform requires various components to ingest data from disparate sources at scale, services to store, manage, process and analyse that data, and interfaces to access and view the data and the analysis results. For example, in the presented use case in Section 4, the devices are the sensors for environmental readings, motion detectors, and noise levels, which generate data in a real-time fashion. In such areas, the proposed implementation can be used. The Fig. 5 details the system components.

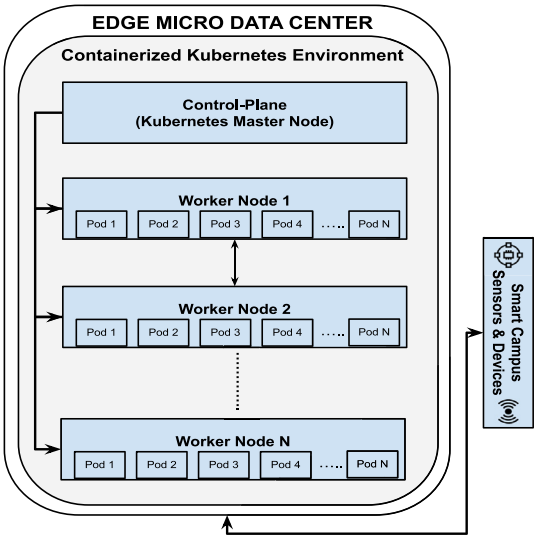


Fig. 4. Edge-based containerised platform design for smart infrastructures and cities.

Storage system: As explained in the previous section, the EMDC consists of multiple worker nodes. Each worker node has a certain storage space available for stateful applications that require persistent storage. To allow the workloads and applications to access the storage dynamically and transparently, a storage system is required that abstracts the storage space and functions from applications and allows consumption through Kubernetes APIs. To meet these requirements, we select Apache Ozone [80] as our storage system. Ozone is a scalable, redundant, and distributed object store that aims to provide a resilient cost–effective solution while addressing the various problems Apache Hadoop faces, such as immutability and small files performance and scalability limitations [81]. We deploy Ozone's management components, i.e., Ozone Manager, Storage Container Manager, Container Storage Interface (CSI) and S3 Gateway, as containers on the EMDC worker nodes. Ozone's datanodes are also deployed as containers, with one datanode container running on each worker node. These datanodes consume the storage space available on the corresponding worker nodes and make it available through the Ozone Manager to the user's client applications. External clients can interact with Ozone through the S3 interface, whereas clients and stateful applications running inside the Kubernetes environment of the EMDC, can interact with Ozone using CSI. CSI is an interface between the container workloads and external storage that supports the creation and configuration of persistent storage on an external storage system, in this case Ozone. Using CSI abstracts simplifies the storage consumption by applications. When an application requests a persistent volume, Ozone seamlessly creates and mounts it into the application's pod and the application can use it as a standard volume, as the management is abstracted and handled by Ozone.

Messaging bus: In real-world deployments, such as our use-case, data comes as streams from diverse sources with different sampling frequencies. This data must be ingested into the platform reliably and fault-tolerantly. Furthermore, our AI/ML models require a highly concurrent and low latency solution to efficiently handle and process large amounts of data in real-time. Apache

Kafka is an open-source distributed event streaming platform that meets our requirements. It is a widely used and tested tool which is capable of building high-performing and reliable data pipelines for mission-critical applications, such as smart transportation, smarthealth care, smart banking, and others [82].

Within our Kubernetes cluster, we deploy three Kafka brokers over three different worker nodes to improve fault tolerance. These brokers are configured with two types of listeners. An internal listener is configured for services and components inside the same EMDC. These components can access Kafka topics and messages using the fully ESO-qualified name of any of the brokers. An external listener is configured for consumers outside the EMDC, e.g. for the other EMDCs in the edge environment. These consumers can access Kafka messages using the IP address and port number of any of the brokers. These listeners are exposed to the network using Kubernetes services that handle various networking aspects, as explained in the previous section. Note that Zookeeper Quorum is required for the functioning of Kafka, and it is also installed as Kubernetes pods with a replication factor of three and is configured using Kubernetes services for access by the Kafka brokers.

As multiple devices and sensors natively use the MQTT protocol to transport messages and readings, an MQTT proxy is configured that provides a scalable and lightweight interface to allow MQTT clients to produce messages directly for Kafka. MQTT Proxy uses a simple mapping scheme of MQTT topics to Kafka topics that are based on regular expressions.[1] This approach avoids the unnecessary duplication of data generated by devices using MQTT clients and provides unified endpoints for the ingestion and consumption of the data.

Analytics engine: Various tools available for processing and analysing large-scale data, such as Apache Solr [83], Apache Spark [84] and Apache Storm [85]. Due to the modular nature of our EMDC, any or multiple of these engines can be used to perform an analysis, depending on the use-case requirements. For this work, the data exploration and analysis has been performed using Apache Spark. The key reasons for choosing Spark are its support for cluster-based computing, faster deployment through ML libraries, and scalable stream processing. We deploy Apache Spark to our platform by using Spark Operator. This allows Spark applications to be defined declaratively and deployed using Kubernetes manifest files. The main application file can be passed to the Spark Docker image specified in the pod specification, along with additional dependencies, and deployed with standard Kubernetes apply commands. The rest of the configuration, execution and management is handled by Spark Operator and Kubernetes, hence making the deployment easy and straightforward for AI/ML developers. When a Spark job is created using Kubernetes Pods, the Spark Operator creates a Spark driver, and when that is up, it automatically requests the creation of Spark Executors and runs the job to completion.

Telemetry and monitoring: In addition to the core functionality mentioned above, other key components are added to the platform to enable smart workload distribution. These include tools for indexing, searching, monitoring and visualising data and results in real time. This is achieved using Elasticsearch, Logstash, and Kibana (ELK) stack. The ELK stack is integrated natively with Kubernetes and performs the aforementioned tasks. When concept drift is detected, these tools are used to find the computing locations that have sufficient resources available and are best suited to run the newly identified tasks, e.g., retraining. Another purpose for such a layer is to log system-level events that can be utilised to develop insights about failures, tasks, security breaches, etc.

[1] MQTT Proxy https://docs.confluent.io/platform/current/kafka-mqtt/intro.html.

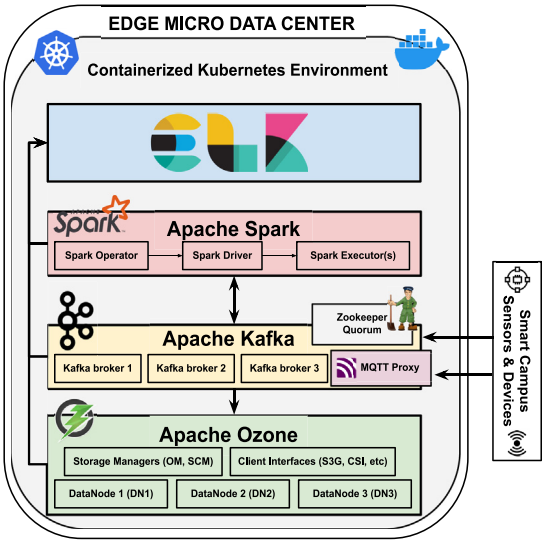


Fig. 5. Components and services deployed in the Edge platform.

4. Use case

At the outset of the Covid-19 pandemic, government policies across the globe were put in place in an effort to limit the mobility of the general population due to the fear of the virus and its consequences for community health. Ranging in severity, exactitude, and duration, confinement policies included lockdowns, telework, school closures, gathering restrictions, and quarantines. As a result, many businesses and institutions worldwide remained closed for several months after the beginning of the pandemic.

The turmoil witnessed in recent years, with the energy crisis, pandemic, and environmental disasters, has exacerbated the need to aim for improved computational sustainability at the edge of the network. For example, McKinsey has called the Covid-19 crisis "technology's Achilles' heel", documenting how machine learning models and algorithms that drive much of the advanced analytics techniques have been affected by the policies introduced during the pandemic, such as lockdowns, travel bans, physical distancing, widespread furloughs, and remote work [86]. For example, modified streets in London contributed unintentionally to an increase of 50% in women's cycling trips in 2020 [5].

The confinement policies had enormous structural consequences on the world economy and on individual lives. According to [3], the Covid-19 pandemic accelerated two notable transformative changes: (i) the virtualisation of all socio-economic activities that enabled telecommunication in large groups, and (ii) a decline in long-distance road trips in favour of short-distance local commuting. These economic activity shifts – from physical to digital and from global to local – resulted in continuously changing patterns in data collected from sensors, leading to the decreasing predictive ability of ML models trained and tested on pre-pandemic datasets.

A case of interest is the fluctuation of carbon dioxide (CO_2) atmospheric concentrations. The emerging consensus in the recent literature is that the reduction in fossil-fuel-based mobility and changes in consumption patterns and energy demand have contributed to a recent worldwide decline in global carbon emissions. In [87], the authors analysed six economic sectors (power, industry, surface transport, public buildings and commerce, residential, and aviation) from more than 69 countries, covering 97% of global CO_2 emissions. The study leveraged energy, daily activity, and policy data as a proxy for measuring country-level

emissions. It concluded that daily fossil CO_2 emissions decreased in 2020 by 17% in comparison to the 2019 levels. The residential sector was the only sector where the mean level of emissions increased during the confinement periods. Most notably, one week of observations taken from smart meter usage data of 115,000 energy customers in the UK (during lockdown) has shown that 30% of households drastically changed their energy consumption patterns compared to the week before the lockdown [88].

At the city level, the variability of anthropogenic emissions of CO_2 is an additional confounding parameter that needs to be accounted for in the analysis of CO_2 fluxes. The authors in [89] considered local emissions of CO_2 from 13 stations in 11 European cities and compared pre-pandemic data with measurements taken in 2020. The study identified a consistent and significant correlation between the severity of confinement policies and reduction in CO_2 emissions across the surveyed sites. Interestingly, the results also show that the magnitude of the recorded CO_2 reductions varies in time and space between cities and areas within the same city.

To address concept drift against the backdrop of the Covid-19 pandemic, Section 3 introduced a novel architecture for smart buildings using the hybrid edge computing paradigm to support the large-scale streaming data acquisition, storage, analysis, and visualisation. To understand the feasibility of this architecture, we verify it by performing predictive modelling in a distributed environment over the developed platform. In our use case, different sensors continuously measure the environmental readings including CO_2, noise levels, and motion counts in a smart campus environment. Given the type of configured devices in the Smart Campus, we can install the proposed EMDC solution in different areas of the environment depending on the number of sensors, computational requirements, sensor distribution, and user applications. For example, as the EMDCs have a small form factor, they can be installed on each level of the building and efficiently handle the data generated from the sensor devices on that level in real time. As a result, such a system may help with low-latency, online learning, data variability concerns, and continuous data monitoring, among other things.

A lot of work has been done in the area of concept drift, with an emphasis on classification algorithms. However, there needs to be more work on addressing concept drift in regression models [13,90]. In addition, very few concept detection methods have been tested against real-world data sets in the context of smart cities [13]. The present use case is among the early efforts where the focus has been on smart buildings and infrastructures. Rest of the details are mentioned in following sub-sections.

4.1. Approach

In this regard, the implemented EMDC architecture was employed in the smart campus environment at Oulu University [8]. We used real-world streaming data from hardwired sensors in a smart campus to forecast CO_2 under the concept drift. The learner forecasting of CO_2 was implemented using an automated feedback-based process for concept drift detection and adaptation which is described in Algorithm 1.

First, we utilised the learner without concept drift detection. We used distributed LSTM. This approach was selected because of its applicability for long-term forecasting. The learner was then enhanced using concept drift detection methods to see whether better results could be achieved. The selected methods were PHT, ADWIN, and KSWIN; see the next subsection for more details. The evaluation of the implemented predictive model, as well as the model integrated with concept drift detection methods, were carried out using Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Square Error (RMSE) methods. Finally, we analysed the results.

4.2. Model

4.2.1. Base learner - LSTM

In this study, we implemented a Long Short-Term Memory (LSTM) method which was proposed to address the gradient disappearing challenge in Recurrent Neural Networks (RNNs) [91]. RNNs have been a popular choice for data-driven time series analysis. The method works sequentially to capture information provided by each time step for predictive analysis. Notably, RNNs are capable of grasping short-term data dependencies. In addition, the data has become more large-scale and real-time, requiring distributed and computationally effective model implementation. A low number of studies are available where different variants of LSTMs have been implemented, with a distributed learning paradigm to address the high computational requirements for centralised learning [92,93]. In this study, the LSTM presented in [13,94] is implemented in a distributed environment leveraging distributed learning for large-scale real-time data streams.

4.2.2. Concept drift detection methods

We implement three state-of-the-art active concept drift detection methods integrated with LSTM using a distributed learning approach. The concept drift detection methods used are PHT, ADWIN, and KSWIN. The methods were shortlisted due to their suitability for time series regression problems.

The **Page–Hinkley** test is a modification of CUSUM, and is primarily used for detecting changes in the data sequentially [69]. It enables efficient change detection by using cumulative differences in data observations and their means sequentially until the current time observation: $C_T = \Sigma_{t=1}^{T}(x_t - \overline{x}_T - \sigma)$, where $\overline{x}_T = \frac{1}{T}\Sigma_{t=1}^{T}x_t$ and σ is a bearable magnitude of change in the data. The minimum m_T is explained as $M_T = min(m_T, t = 1...T)$. PHT computes the difference between M_T and m_t. If the computed difference is greater than a user-defined threshold (λ) that is passed as an input parameter (allowed magnitude of change), an alarm is generated indicating concept drift detection [69]. The method has been found to be suitable for abrupt drift scenarios [13,95].

Adaptive windowing has been proposed to detect concept drift in data using a windowing approach, where the input sequence $x_0, x_1, x_2....x_n$ is bounded between [0,1] by re-scaling if the lower and upper bound of data sequence exceeds the boundary [13,70]. ADWIN uses two windows, one fixed window with known averages of the data in the sequence, and the other with unknown means from newly arriving data. If the subset of two large windows shows a distinct mean, the method signals that the values in the windows are different, while discarding the old sub-window. This is done using a Hoeffding bound $\epsilon_c = \sqrt{\frac{1}{2m}.ln\frac{4|W|}{\delta}}$, where the length of the window is denoted by $|W|$, m is defined as the harmonic mean of $|W_0|$, $|W_1|$, and δ represents user-defined confidence value (0,1) to evaluate if both windows relate to same distribution (recommended value: 0.2) [69].

Kolmogorov–Smirnov windowing is a concept drift detection method developed based on the non-parametric Kolmogorov–Smirnov test. The method accepts one dimensional data without the need for underlying assumptions of the data distribution. The data is monitored by comparing the distance between two empirical cumulative distributions [96]. In KSWIN, a sliding window ψ of fixed size n is maintained and the last samples of size r are considered to represent the latest statistically valid concept as R. The last samples r are taken from the initial $n - r$ data samples of ψ to constitute the last approximate concept W, later the Kolmogorov–Smirnov test is performed on R and W of similar sizes. According to [96] the size of window r and the confidence level α should be kept smaller to avoid the false positive in the case of streaming context. The concept drift in KSWIN is detected if: $dist(R, W) > \sqrt{-\frac{ln\alpha}{r}}$.

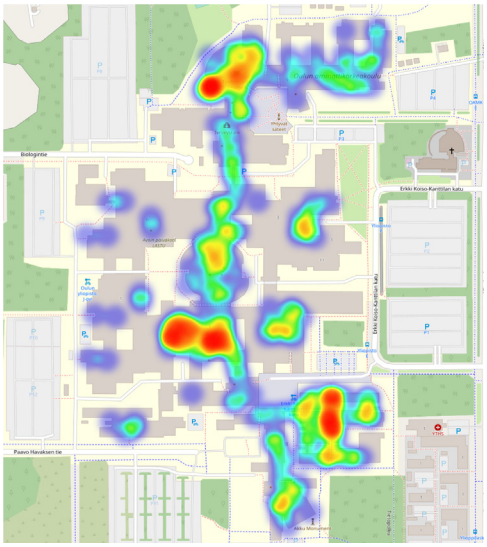


Fig. 6. Carbon dioxide footprint in different areas of the Smart Campus.

4.3. Use case implementation

Dataset. The implementation of the distributed time series learner integrated with concept drift detection methods was carried out using real-world streaming data from a smart campus in Finland (University of Oulu) [8]. The smart campus has about 419 sensors measuring different parameters such as CO_2, temperature, noise, movements, humidity, etc. In this study, we use streaming data containing CO_2 measurements from 5 different open space locations for collaborative work, studies, and meetings called the Tellus Innovation Arena (Tellus), where every hour hundreds of student pass through and study 24/7. The five sensors highlighted in Fig. 7 were selected due to spatio-temporal dependencies in the measurements (each area can have its own patterns). Secondly, the chosen sensor locations were close to entrance points with access to the main seating area. These sensors were closely situated by forming a grid, leading to fewer deviations in measurements and closer trends, resulting in a suitable setup for predictive modelling. For this use case, the historical data from June 2020 till January 2022 were used in a streaming manner. The data contains the location of sensors (Long/Lat), timestamp, device ID, and CO_2 measurements. An example of the CO_2 footprint in the university campus at a particular time is shown as a heat map in Fig. 6. For more details, please refer to this URL.[2]

Standalone LSTM. The distributed LSTM was implemented using the edge-based infrastructure proposed in this study, see Fig. 5. The implementation used Apache Spark (Pyspark) by inheriting its functionalities, such as pipelines and estimators. For a distributed implementation, the learner is configured with three workers in the pipeline and one master node that triggers and maintains the learning job.

LSTM with concept drift methods. We provide a novel algorithmic approach to implement the use case given in this study. The time series learner (i.e. LSTM) is trained with historical data using the distributed learning paradigm with Apache Spark. The trained learner uses historical data first and streaming data at a later stage to compute predictions and detect concept drift using the PHT, ADWIN and KSWIN methods. A message is broadcasted using Apache Kafka[3] as a communication bus for assigned Kafka topics (q_{t1} & q_{t2}) for detected concept drift, exception messages, and if the data stream has no values. If concept drift is detected, the Kafka producer kp_1 generates an alert stating "concept drift has been detected", consumed by Kafka consumer kc_1 to update the model and save the learner's current state. In addition, if the data stream is empty, kp_1 broadcasts an exception message and the process is started again. However, if no concept drift is detected, Kafka producer kp_2 broadcasts a message for model deployment, and kc_2 deploys the model. Algorithm 1 and Fig. 8 provide details of the implemented approach.

Algorithm 1 Feedback-based Concept drift detection and adaptation process

Historical data to train hd_T Streaming data ds_x, Target Variable ds_t, Predictions ds_o, Learner l_n, Worker Node W_n
Kafka topic for concept drift alert q_{t1}, Kafka topic for exceptions and messages q_{t2}, Kafka consumer for $q_{t1} \leftarrow kc_1$, Kafka consumer to deploy model kc_2
Kafka producer for $q_{t1} \leftarrow kp_1$, Kafka producer to publish model kp_2
Initialise l_n
Train $l_n \leftarrow hd_T$ with W_{n-1}
while $ds_x = True$ **do**
  for each ds_x **do**
    Compute $ds_o \leftarrow l_n$
  end for
  if len($ds_o > 0$) **then**
    Call Concept drift detector ▷ Kolmogorov Smirnov
    Compute Error (ds_t, ds_o)
    if CD is True **then**
      $q_{t1} \leftarrow kp_1$
      Alert is sent
      Update model $\leftarrow kc_1$
      $kp_2 \leftarrow l_n$
      $kc_2 \leftarrow q_{t2}$
      $hd_t + ds_o = hd_t$
    else
      $kp_2 \leftarrow l_n$
      $kc_2 \leftarrow q_{t2}$
      $hd_t + ds_o = hd_t$
    end if
  else
    $qt_2 \leftarrow kp_1$ (No values available)
  end if
end while

5. Results

This section discusses the findings of the smart campus experiment on predictive analytics employing the distributed learning paradigm for concept drift detection using the EMDC architecture, see Sections 3.3 and 4.

Results from a standalone LSTM. The CO_2 data from the smart campus was analysed to determine adequate parameters for the learner implementation, for example, trends and seasonality. After outliers were removed, the dataset was divided into training and test sets.

LSTM with a linear output layer from [13] was adapted for this scenario implementation. The data was first scaled between [0,1] and further transformed with sequences of 100 time steps, shifted by a single time step. The training was done by opting for Adam's optimiser synchronously with a learner rate of 0.001 and epochs set as 25.

Fig. 9 presents the learner output which shows that the model was able to follow the original observations quite well. However,

[2] Smart Campus https://smartcampus.oulu.fi/manage/.
[3] Apache Kafka https://kafka.apache.org/.

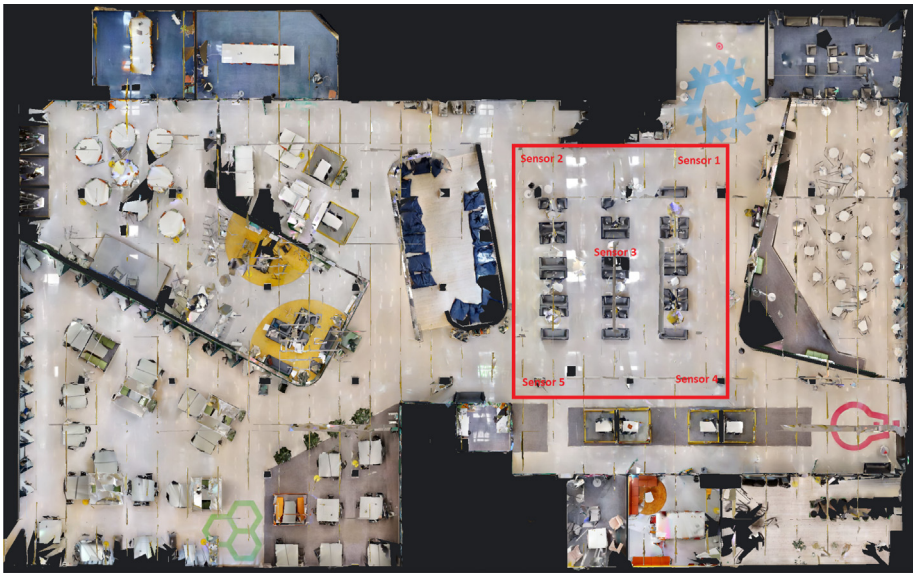


Fig. 7. Map of Tellus sensors used for CO_2 measurements.
Source: Adapted from [97].

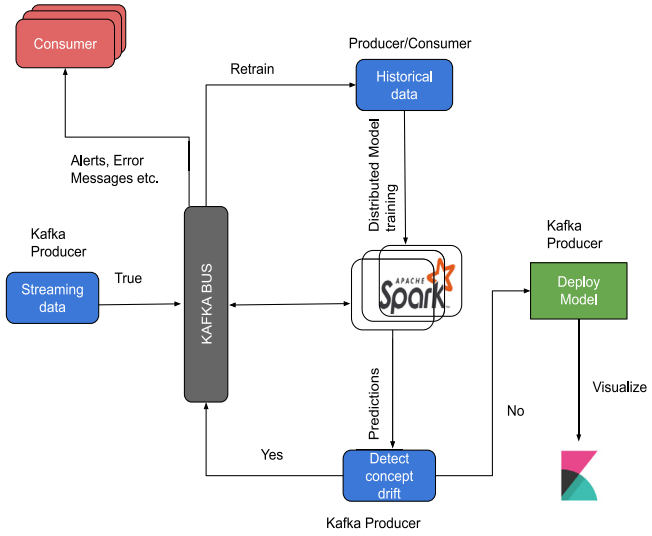


Fig. 8. General flow of introduced algorithmic process.

with the time bracket of December 2021 until January 2022, the model was unable to perform accurately due to a possible change in the trend. Additionally, if we examine the Table 2, the performance was satisfactory, considering the minimum observation to be 341.9 ppm and the maximum to be 1283.0 ppm.

Results from LSTM with integrated concept drift detection methods. The finalised model described in the previous section was then further utilised to integrate with concept drift detection methods. The dataset used in this study is from a real-world university campus facility where we observed that the data could be influenced by different factors such as the holiday season. It was hypothesised that, *"The data may incorporate a concept drift due to the university holidays and remote teaching during Covid 19, which may influence the prediction. Therefore, integrating concept drift detection methods could potentially enhance the learners' predictive performance when deployed in such environments"*.

The implementation comprises three concept drift detection methods integrated with distributed LSTM with Apache Spark. After experimenting with several setups, the concept drift detection techniques were employed with the following parameters: (i) for KSWIN $\alpha = 0.001$, number of instances $= 100$, and seed $= 50$, (ii) for ADWIN $\delta = 0.005$, (iii) for PHT $\alpha = 1$, $\delta = 25.0$, and number of instances $= 100$.

Table 2
Evaluation of the learner time series with integrated concept drift detection methods, where the min. observation is 341.9 and the max. is 1283.0.

Approach	MAE	MAPE	RMSE
LSTM	42.8	8.5	59.2
LSTM with PHT	24.02	4.8	33.56
LSTM with ADWIN	23.91	4.78	32.22
LSTM with KSWIN	19.8	3.88	27.9

Fig. 9 shows the predicted values without considering concept drift. Fig. 11 shows various concept drifts detected by the implemented algorithms. Table 2 proves our hypothesis and shows that ignoring concept drifts results in larger errors, and the implemented algorithms improve the performance through concept drift detection. Based on the evaluation of the results shown in Table 2, the KSWIN fared well compared to the others. However, the detection of concept drift in the generated prediction from the learner was sequential in the case of KSWIN. On the other hand, PHT and ADWIN were able to detect concept drift in more instances.

Currently, the size of the test data was small as the setup on the campus was relatively new and continuously undergoing expansion. Due to the availability of more patterns and trends in larger datasets, a larger dataset would make the learner more effective and capable of detecting different types of concept drifts in the data. Before the deployment of the model for real-world usage, more data is therefore required.

6. Discussion

In the last few years, technological advances have led to many innovative initiatives to fulfil the goals for sustainable ecosystems, e.g. in educational organisations, cities, governance institutions, and industry. The prevalence of booming technologies such as IoT, 5G, and edge computing has resulted in the availability of enormous amounts of raw data for processing,

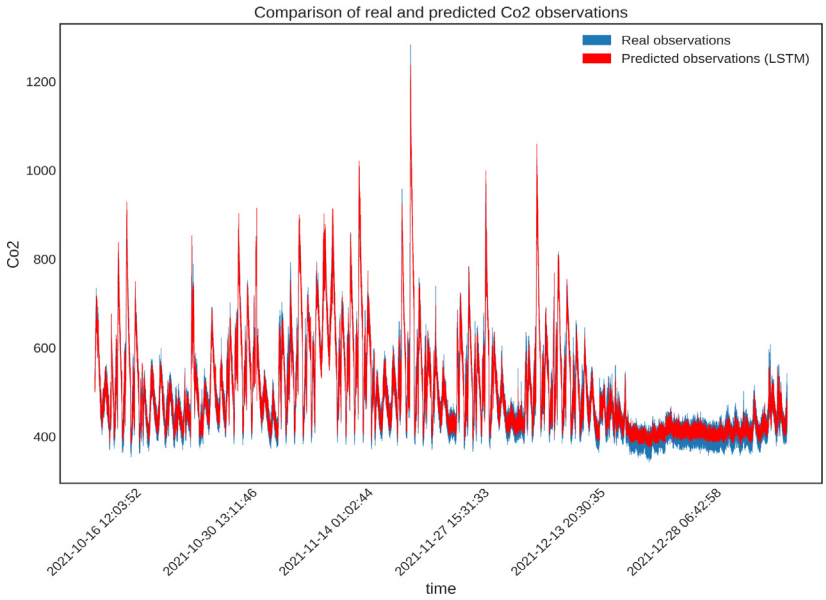


Fig. 9. Comparison of real and predicted CO_2 observations.

predominantly in smart city ecosystems. Technocrats, analysts, decision-makers, and researchers are attempting to harness the large-scale data for the betterment of society and to make the global ecosystem sustainable. Therefore, in this context, more efficient, cost-effective, and fault-tolerant edge computing-based solutions are needed to model and monitor the continuously arriving data at the edge.

6.1. EMDC proposal

This paper presents a novel solution called EMDC to fulfil the goals of data collection, processing, analysis, and delivery to the end stakeholders in a hybrid cloud–edge continuum. The proposed architecture combines the benefits of the cloud and edge in terms of higher computing power and lower latencies. The design allows flexible decoupling of applications and services from the underlying infrastructure. It also seamlessly incorporates various smart city use cases and applications. By monitoring the platform using analytics and tracking resource consumption, it can predict workload resource needs, making it easier to migrate workloads between cloud and edge environments. This is further exploited with the novel application of a learner integrated with concept drift detection methods. In addition, the resource requirements are determined using the telemetry layer and post-concept drift detection (if data at a particular location requires re-training or resampling), so that the workloads can be assigned to the best computing locations, supported by live migration functionality.

The hybrid cloud edge computing continuum architecture also enables smart cities to scale their infrastructure up or down depending on their needs. This makes it easier to manage fluctuating workloads and ensure that resources are being used efficiently. The telemetry and monitoring components of the proposed architecture further help with the scalability and reduced latency by exploiting real-time data insights into the performance, health, and availability of the continuum resources.

The proposed solution is based on a completely containerised environment and is managed using Kubernetes. Given the distributed nature of the edge environments and the varying application requirements, containerisation presents a superior virtualisation approach to traditional virtual machines or bare metal implementations. Packaging applications in containers makes them portable as all the dependencies are included within the package. Containerised environments are resource efficient as resources are allocated at a more granular level and can be scaled as required. These benefits are leveraged in the implemented architecture for the presented use case by running our workloads as containers; see Fig. 10. Finally, orchestration through Kubernetes makes our solution agile and easy to manage as various key operational functions such as fault-tolerance, load-balancing, and scaling are separated from the application logic and handled by the orchestrator instead.

Fault-tolerance is one of the key requirements for any distributed platform regardless of the technology paradigm used. In our proposal (see Fig. 5), the continuous monitoring of the solution is ensured by the telemetry layer where all system events are logged to contain the possible failures. The data collection and processing are done with a combination of Apache Kafka and Apache Spark, and the failures and any process to be re-initiated are maintained by logging the state of the jobs.

Real-world data can suffer from completeness issues, outliers, and other rule violations that are necessary to address at the system level on the go. EMDC ensures data schema validation. For example, the sensor devices could be integrated with the EMDC using an MQTT and Apache Kafka broker, where the concepts of schema validation and checkpoints can be leveraged to address such issues.

6.2. Concept drift

The data from smart cities and smart buildings is now available in streaming fashion. However, in non-stationary environments, the characteristics of the data are continuously drifting. Concept drift has therefore become an essential challenge to be mitigated in the context of smart cities or dynamic environments where applications of machine learning are evident.

In addition to our novel EMDC solution, we address challenges related to dynamic smart scenarios. In this paper, we provide a real-world use case addressing the problem of concept drift. In our use case, we experiment with three different concept drift detection methods (PHT, ADWIN, and KSWIN) using edge-deployed big data technology solutions such as Apache Kafka, Apache Spark, Apache Ozone, and ELK.

The implementation of concept drift detection methods in distributed edge computing environments poses many challenges.

```
ubuntu@OULU----:~$ kubectl describe pod workload.spark
Name:         workload.spark
Namespace:    default
Node:         oulu2-work01/10.13.2.2
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"v1","kind":"Pod","metadata"
                :{"annotations":{},"name":"workload.spark"
                ,"namespace":"default"},"spec":{"contai...
Status:       Running
IP:           10.244.1.14
IPs:
  IP:  10.244.1.14
Containers:
  workload:
    Container ID:   docker://e279b19b4fa1216ac98af6c2ea0b8510
                    e2e535f3126de4e7c24f6edc749935d0
    Image:          10.13.2.1:5000/workload/spark
    Image ID:       docker-pullable://10.13.2.1:5000/workload/
                    spark@sha256:dc70e
                    ba897ca8d311fba765bfd1a7e8eedf7f42d42a4502
                    2e310e1ccaaed1d0
    Mounts:         /var/run/secrets/kubernetes.io/serviceaccount
                    from default-token-xkhkj (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-xkhkj:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-xkhkj
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s

Events:
  Type    Reason     Age        From                    Message
  ----    ------     ----       ----                    -------
  Normal  Scheduled  <unknown>                          Successfully assigned
                                                        default/workload.spark
                                                        to oulu2-work01
  Normal  Pulling    3m4s       kubelet, oulu2-work01   Pulling image
                                                        "10.13.2.1:5000
                                                        /workload/spark"
  Normal  Pulled     2m18s      kubelet, oulu2-work01   Successfully pulled
                                                        image "10.13.2.1:5000
                                                        /workload/spark" in
                                                              46.473354841s
  Normal  Created    2m6s       kubelet, oulu2-work01   Created container
                                                        workload
  Normal  Started    2m6s       kubelet, oulu2-work01   Started container
                                                        workload
```

Fig. 10. Event logs from executed Spark job in containerised EMDC.

First, the integration of these methods with the learners can be troublesome in some cases when the data is improperly balanced between all the edge nodes. For example, the wait time for the concept drift detection method results may increase due to the uneven distribution of data across the nodes. Therefore, a scheduled task can be delayed or result in failure. This study configures the exceptions and delay errors with Apache Kafka to address them in real time. Kafka consumer continuously listens to the configured topic to generate an alert for any issues. Moreover, in the presented use case, the state of the implemented algorithm is stored in the system, thereby enabling the mitigation and re-initiation of the tasks, see Algorithm 1. The proposed algorithm enables the continuous monitoring of learners and detects concept drift using Kafka topics and telemetry tools integrated into the platform. Such an approach can benefit in moving workloads towards particular nodes, e.g. if resampling is required post-concept drift detection for node-specific data or a subset of the data from a particular edge node for detected concept drift. Similarly, if there are load balancing issues and one edge node has skewed data, telemetry logs from the system and data ingestion jobs can support finding the root cause.

In addition, the pivotal requirement for streaming analytics is to ensure the correct parameters for tuning and selecting the models to handle any drifting behaviours in the streaming data. However, in dynamic environments such as smart buildings, the environmental measurements change very quickly, and the trend in the last three months may be no longer valid due to holidays, construction work, and others. Therefore, real-world drifting situations differ from simulated data drifts and require prior knowledge of the data. For example, the ideal measurement of CO_2 in indoor environments is 400 ppm, and in larger areas, 800 ppm [98]; in the presented use case the values increased to 6000 ppm due to malfunctioning devices exhibiting the concept drift phenomena. The proposed concept drift detection framework can support monitoring the intrinsic characteristics of data and facilitation of the possibilities of concept drift detection and adaptation. However, it can require periodic manual intervention to set thresholds and tune the parameters of the learners.

Finally, the learners integrated with concept drift detection methods require adaptation strategies to be in place for effective forecasting. In this study, we have implemented a feedback-based active concept drift detection approach using real-world streaming data. The retraining process in real-world settings cannot be initiated without prior knowledge of the type of concept drift, the amount of data required for model training, and false positive tests. In addition, if data deviations are small for detected concept drift, data resampling can be performed before retraining the model. Much of the existing literature proposes concept drift detection and approaches. However, these methods are mainly exploited using synthetic and real-world static datasets [99–101], unlike in the case of real-world data streams where changes in data distribution are often expected, and guaranteeing the correctness of the concept drift is a great challenge. Several approaches are available to ensure the correctness of detected concept drift. The drifts can be distinguished either by hypothesis testing or continuously monitoring the defined thresholds for warning and detection zones in the context of active detection methods. In the case of passive approaches, ensemble methods are widely used to address these challenges.

However, in the passive approaches, the detection of concept drift is not needed, and the learner is updated based on a predefined frequency. However, this method can be computationally exhaustive due to unnecessary retraining. In addition, it allows the detected concept drift to be studied, i.e. when and why the drift was detected. To study why a concept drift occurred, hypothesis testing is needed with prior knowledge of data patterns. In our opinion, such an approach is efficient, enabling the learner to retrain only when it is needed.

7. Conclusions and limitations

In this study, we presented a novel containerised EMDC architecture for smart cities to support computationally demanding data analysis and storage, enable integration with different smart city use cases, and facilitate on-demand live migration of workloads based on telemetry tools and resource requirements in a hybrid cloud edge continuum. The EMDC architecture was deployed at the University of Oulu Smart Campus, and the feasibility of the proposed design was evaluated by conducting experiments on this platform. The implemented solution comprised an innovative algorithm to perform predictive modelling and detect concept drift in real-world data streams. We demonstrated that in smart environments such as the present use case of the smart campus, integrating concept drift detection algorithms can enhance the effectiveness of predictions drawn from a learner. However, in non-stationary environments, the data patterns are continuously changing. The required learner parameters should therefore be selected carefully. In such cases, concept drift detection algorithms are well suited to reducing manual intervention for tuning parameters and retraining. If there is a significant change in the data, an alarm is generated, and learners are retrained.

The current setup has some limitations. First, to explore the generality of the proposed solution, we need to explore its use in other cases related to smart cities such as smart mobility or smart living. Second, the data governance in the proposed EMDC

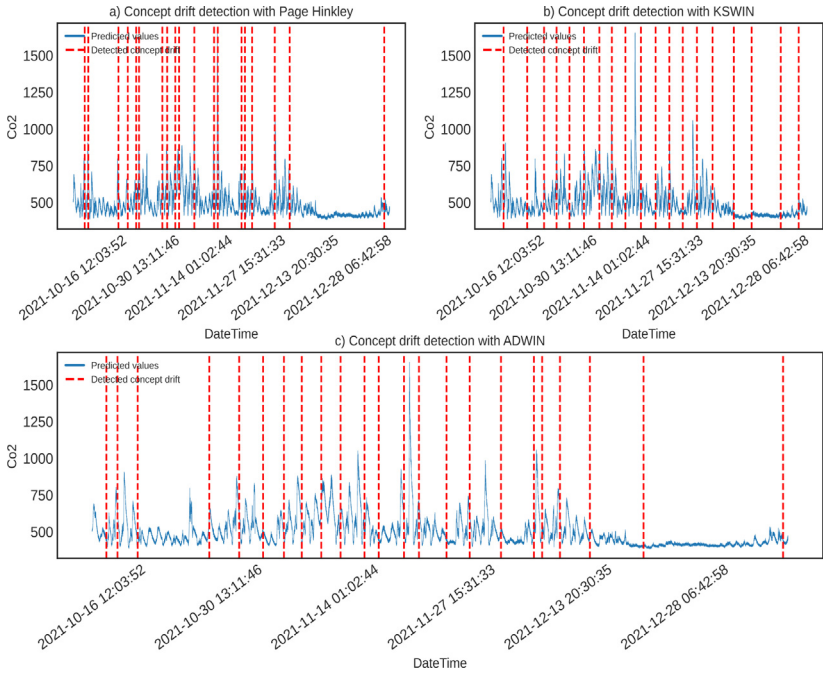


Fig. 11. Comparison of concept drift on predicted CO_2 observations using different methods.

solution can be improved by introducing data management tools such as Apache Ranger [102] and Apache Atlas [103]. While the proposed modular infrastructure allows for the seamless integration of such tools, they were beyond the scope of this study and were not deployed at this stage. These features will be explored in future versions of EMDC deployment in the smart campus.

It was further observed that the proposed algorithm worked well in the presented use case. Naturally, some integrated algorithms performed better than others, fitting the situation better. In this use case, a combination of LSTM with KSWIN had a decreased MAPE compared to the others. However, a larger dataset in the future could help in fine tune the algorithm. Future work therefore requires more experimentation and false-positive tests to enable the model to be operational in a smart environment.

CRediT authorship contribution statement

Hassan Mehmood: Conceptualization, Methodology, Architecture, Data curation, Software, Validation, Formal analysis, Writing – original draft, Writing – review & editing. **Ahmed Khalid:** Architecture, Writing – original draft, Writing – review & editing. **Panos Kostakos:** Writing – original draft, Writing – review & editing. **Ekaterina Gilman:** Writing – review & editing. **Susanna Pirttikangas:** Writing – review & editing.

Declaration of competing interest

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

Data availability

Data will be made available on request.

Acknowledgements

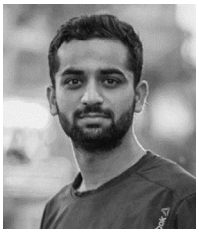
This study was funded by the Academy of Finland, Finland (grants 337614, 323630, 318927), European Commission grants IDUNN (grant no. 101021911), ECSEL Joint Undertaking under grant agreement No 876967, and KDT JU under grant agreement No. 101097560 (CLEVER project). All authors have read and agreed to the published version of the manuscript.

References

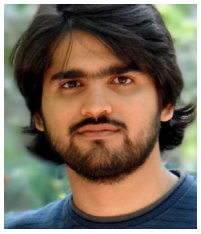
[1] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: Vision and challenges, IEEE Internet Things J. 3 (5) (2016) 637–646.
[2] S.A. Hossain, M.A. Rahman, M.A. Hossain, Edge computing framework for enabling situation awareness in IoT based smart city, J. Parallel Distrib. Comput. 122 (2018) 226–237.
[3] A. Newman, et al., Covid, cities and climate: historical precedents and potential transitions for the new economy, Urban Sci. 4 (3) (2020) 32.
[4] UN, World Urbanization Prospects: The 2018 Revision, United Nations New York, NY, USA, 2018.
[5] D.J. Hill, M. Acuto, Parklets, Traffic-Free Zones and Outdoor Eating: How COVID Is Transforming Our Cities, World Economic Forum, 2022, URL https://www.weforum.org/agenda/2022/01/traffic-free-zones-outdoor-eating-covid-transform-cities/.
[6] Smart Building Market Size, Share and COVID-19 Impact Analysis, Fortune Business Insights, 2021, URL https://www.fortunebusinessinsights.com/industry-reports/smart-building-market-101198.
[7] Frost & Sullivan, Future of smart and connected homes, forecast to 2025, 2019, URL https://store.frost.com/future-of-smart-and-connected-homes-forecast-to-2025.html.
[8] E. Gilman, S. Tamminen, R. Yasmin, E. Ristimella, E. Peltonen, M. Harju, L. Loven, J. Riekki, S. Pirttikangas, Internet of things for smart spaces: A university campus case study, Sensors (ISSN: 1424-8220) 20 (13) (2020) http://dx.doi.org/10.3390/s20133716, URL https://www.mdpi.com/1424-8220/20/13/3716.
[9] H. Mehmood, E. Gilman, M. Cortes, P. Kostakos, A. Byrne, K. Valta, S. Tekes, J. Riekki, Implementing big data lake for heterogeneous data sources, in: 2019 Ieee 35th International Conference on Data Engineering Workshops, Icdew, IEEE, 2019, pp. 37–44.
[10] A. Mishra, A.V. Jha, B. Appasani, A.K. Ray, D.K. Gupta, A.N. Ghazali, Emerging technologies and design aspects of next generation cyber physical system with a smart city application perspective, Int. J. Syst. Assur. Eng. Manag. (2022) 1–23.

[11] N. Mohamed, J. Al-Jaroodi, S. Lazarova-Molnar, I. Jawhar, Applications of integrated IoT-fog-cloud systems to smart cities: A survey, Electronics 10 (23) (2021) 2918.
[12] N.A. Sulieman, L. Ricciardi Celsi, W. Li, A. Zomaya, M. Villari, Edge-oriented computing: A survey on research and use cases, Energies 15 (2) (2022) 452.
[13] H. Mehmood, P. Kostakos, M. Cortes, T. Anagnostopoulos, S. Pirttikangas, E. Gilman, Concept drift adaptation techniques in distributed environment for real-world data streams, Smart Cities 4 (1) (2021) 349–371.
[14] T. Lähderanta, T. Leppänen, L. Ruha, L. Lovén, E. Harjula, M. Ylianttila, J. Riekki, M.J. Sillanpää, Edge computing server placement with capacitated location allocation, J. Parallel Distrib. Comput. 153 (2021) 130–149.
[15] H. Kokkonen, L. Lovén, N.H. Motlagh, J. Partala, A. González-Gil, E. Sola, I. Angulo, M. Liyanage, T. Leppänen, T. Nguyen, et al., Autonomy and intelligence in the computing continuum: Challenges, enablers, and future directions for orchestration, 2022, arXiv preprint arXiv:2205.01423.
[16] G. Tancev, Relevance of drift components and unit-to-unit variability in the predictive maintenance of low-cost electrochemical sensor systems in air quality monitoring, Sensors 21 (9) (2021) 3298.
[17] R. Xu, Y. Cheng, Z. Liu, Y. Xie, Y. Yang, Improved long short-term memory based anomaly detection with concept drift adaptive method for supporting IoT services, Future Gener. Comput. Syst. 112 (2020) 228–242.
[18] M. Lima, M. Neto, T. Silva Filho, R.A.d.A. Fagundes, Learning under concept drift for regression—A systematic literature review, IEEE Access 10 (2022) 45410–45429.
[19] L. Wang, Y. Zhang, X. Zhu, Concept drift-aware temporal cloud service APIs recommendation for building composite cloud systems, J. Syst. Softw. 174 (2021) 110902.
[20] M. Jain, G. Kaur, Distributed anomaly detection using concept drift detection based hybrid ensemble techniques in streamed network data, Cluster Comput. (2021) 1–16.
[21] F. Carcillo, A. Dal Pozzolo, Y.-A. Le Borgne, O. Caelen, Y. Mazzer, G. Bontempi, Scarff: a scalable framework for streaming credit card fraud detection with spark, Inf. Fusion 41 (2018) 182–194.
[22] S. Disabato, M. Roveri, Tiny machine learning for concept drift, 2021, arXiv preprint arXiv:2107.14759.
[23] Y. Yang, S. Ding, Y. Liu, S. Meng, X. Chi, R. Ma, C. Yan, Fast wireless sensor for anomaly detection based on data stream in an edge-computing-enabled smart greenhouse, Digit. Commun. Netw. 8 (4) (2022) 498–507.
[24] H. Luo, H. Cai, H. Yu, Y. Sun, Z. Bi, L. Jiang, A short-term energy prediction system based on edge computing for smart city, Future Gener. Comput. Syst. 101 (2019) 444–457.
[25] L.U. Khan, I. Yaqoob, N.H. Tran, S.A. Kazmi, T.N. Dang, C.S. Hong, Edge-computing-enabled smart cities: A comprehensive survey, IEEE Internet Things J. 7 (10) (2020) 10200–10232.
[26] M. Lillstrang, M. Harju, G. del Campo, G. Calderon, J. Röning, S. Tamminen, Implications of properties and quality of indoor sensor data for building machine learning applications: Two case studies in smart campuses, Build. Environ. 207 (2022) 108529.
[27] J.M. Corchado, P. Chamoso, G. Hernández, A.S.R. Gutierrez, A.R. Camacho, A. González-Briones, F. Pinto-Santos, E. Goyenechea, D. Garcia-Retuerta, M. Alonso-Miguel, et al., Deepint. net: A rapid deployment platform for smart territories, Sensors 21 (1) (2021) 236.
[28] N.E. Klepeis, W.C. Nelson, W.R. Ott, J.P. Robinson, A.M. Tsang, P. Switzer, J.V. Behar, S.C. Hern, W.H. Engelmann, The national human activity pattern survey (NHAPS): a resource for assessing exposure to environmental pollutants, J. Expo. Sci. Environ. Epidemiology 11 (3) (2001) 231–252.
[29] A. Kaneko, N. Sugino, T. Suzuki, S. Ishijima, A step towards the smart campus: a venture project based on distance learning by a hybrid video conferencing system, in: Smc 2000 Conference Proceedings. 2000 Ieee International Conference on Systems, Man and Cybernetics.'Cybernetics Evolving To Systems, Humans, Organizations, and their Complex Interactions' (Cat. No. 0), Vol. 1, IEEE, 2000, pp. 38–43.
[30] L. Tan, N. Wang, Future internet: The internet of things, in: 2010 3rd International Conference on Advanced Computer Theory and Engineering, Vol. 5, ICACTE, IEEE, 2010, pp. V5–376.
[31] T. Anagnostopoulos, P. Kostakos, A. Zaslavsky, I. Kantzavelou, N. Tsotsolas, I. Salmon, J. Morley, R. Harle, Challenges and solutions of surveillance systems in IoT-enabled smart campus: A survey, IEEE Access 9 (2021) 131926–131954.
[32] B.N. Silva, M. Khan, C. Jung, J. Seo, D. Muhammad, J. Han, Y. Yoon, K. Han, Urban planning and smart city decision management empowered by real-time data processing using big data analytics, Sensors 18 (9) (2018) 2994.
[33] C. Perera, Y. Qin, J.C. Estrella, S. Reiff-Marganiec, A.V. Vasilakos, Fog computing for sustainable smart cities: A survey, ACM Comput. Surv. 50 (3) (2017) 1–43.
[34] H. Mehmood, M. Hiltunen, T. Makkonen, M. Immonen, S. Pirttikangas, R. Heikkilä, Road map for implementing AI-driven oulu smart excavator, in: Proceedings of the 38th International Symposium on Automation and Robotics in Construction, ISARC 2021, Dubai, United Arab Emirates, November 2-4, 2021, International Association for Automation and Robotics in Construction, 2021.
[35] Frost & Sullivan. Smart City Adoption Timeline, Global Information, Inc., Frost & Sullivan, 2018, URL https://www.giiresearch.com/report/fs604427-smart-city-adoption-timeline.html.
[36] T. Anagnostopoulos, P. Kostakos, I. Salmon, Y. Psaromiligkos, K. Ntalianis, S.J. Ramson, Spatiotemporal authentication system architecture for smart campus safety, in: 2022 4th International Conference on Smart Sensors and Application, ICSSA, IEEE, 2022, pp. 155–160.
[37] A.S. Syed, D. Sierra-Sosa, A. Kumar, A. Elmaghraby, IoT in smart cities: A survey of technologies, practices and challenges, Smart Cities 4 (2) (2021) 429–475.
[38] A. Pandya, P. Kostakos, H. Mehmood, M. Cortes, E. Gilman, M. Oussalah, S. Pirttikangas, Privacy preserving sentiment analysis on multiple edge data streams with apache NiFi, in: 2019 European Intelligence and Security Informatics Conference, EISIC, IEEE, 2019, pp. 130–133.
[39] F. Almalki, S.H. Alsamhi, R. Sahal, J. Hassan, A. Hawbani, N. Rajput, A. Saif, J. Morgan, J. Breslin, et al., Green IoT for eco-friendly and sustainable smart cities: future directions and opportunities, Mob. Netw. Appl. (2021) 1–25.
[40] The action plan for 2020. URL https://www.seisakukikaku.metro.tokyo.lg.jp/en/basic-plan/actionplan-for-2020/,
[41] M. Morningstar, Tokyo, 2021, About Smart Cities®. URL https://www.aboutsmartcities.com/smart-city-tokyo/.
[42] M. Jang, S.-T. Suh, U-city: new trends of urban planning in Korea based on pervasive and ubiquitous geotechnology and geoinformation, in: International Conference on Computational Science and Its Applications, Springer, 2010, pp. 262–270.
[43] A. Ekman, Smart cities: Chinese ambitions in the time of coronavirus, Politique Etrangere (3) (2020) 141–151.
[44] Y. Yu, N. Zhang, Does smart city policy improve energy efficiency? Evidence from a quasi-natural experiment in China, J. Clean. Prod. 229 (2019) 501–512.
[45] R. Hu, The state of smart cities in China: The case of Shenzhen, Energies 12 (22) (2019) 4375.
[46] Smart America: Smart Cities USA. Smart America RSS2. URL https://smartamerica.org/teams/smart-cities-usa/.
[47] General Assembly of the European Innovation Partnership on Smart Cities and Communities (EIP-SCC), European Commission - European Commission, 2021, URL https://smart-cities-marketplace.ec.europa.eu/news-and-events/events/2018/eip-scc-general-assembly-2018-0.
[48] L. Sanchez, L. Muñoz, J.A. Galache, P. Sotres, J.R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis, et al., SmartSantander: IoT experimentation over a smart city testbed, Comput. Netw. 61 (2014) 217–238.
[49] CUTLER. URL https://www.cutler-h2020.eu/.
[50] A.P. Chalikias, I. Tsampoulatidis, F. Tsalakanidou, S. Nikolopoulos, I. Kompatsiaris, N. Komninos, K. Doudouliakis, G. Papastergios, P. Papafilis, S. Karkaletsi, et al., Evidence-driven policy-making using heterogeneous data sources—The case of a controlled parking system in Thessaloniki, Data & Policy 2 (2020).
[51] G. Pantalona, F. Tsalakanidou, S. Nikolopoulos, I. Kompatsiaris, F. Lombardo, D. Norbiato, M. Ferri, L. Kovats, H. Haberstock, Decision support system for flood risk reduction policies: The case of a flood protection measure in the area of Vicenza, Data & Policy 3 (2021).
[52] Frost & Sullivan, Digital Trends in Education–2022 Investment Plans Address Talent Shortage and Remote Learning Adapting to Meet the Needs of Modern Students, in: Voice of Customer, (no. K69E / 00) Frost & Sullivan, 2022.
[53] P. Bellini, P. Nesi, M. Paolucci, I. Zaza, Smart city architecture for data ingestion and analytics: Processes and solutions, in: 2018 IEEE Fourth International Conference on Big Data Computing Service and Applications, BigDataService, IEEE, 2018, pp. 137–144.
[54] B. Cheng, S. Longo, F. Cirillo, M. Bauer, E. Kovacs, Building a big data platform for smart cities: Experience and lessons from santander, in: 2015 IEEE International Congress on Big Data, IEEE, 2015, pp. 592–599.

[55] C. Badii, E.G. Belay, P. Bellini, M. Marazzini, M. Mesiti, P. Nesi, G. Pantaleo, M. Paolucci, S. Valtolina, M. Soderi, et al., Snap4city: A scalable iot/ioe platform for developing smart city applications, in: 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation, SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI, IEEE, 2018, pp. 2109–2116.
[56] Snap4City. URL https://www.snap4city.org.
[57] F. Cicirelli, A. Guerrieri, G. Spezzano, A. Vinci, An edge-based platform for dynamic smart city applications, Future Gener. Comput. Syst. 76 (2017) 106–118.
[58] G. Vítor, P. Rito, S. Sargento, F. Pinto, A scalable approach for smart city data platform: Support of real-time processing and data sharing, Comput. Netw. 213 (2022) 109027.
[59] H. Xu, A. Berres, S.B. Yoginath, H. Sorensen, P.J. Nugent, J. Severino, S.A. Tennille, A. Moore, W. Jones, J. Sanyal, Smart mobility in the cloud: Enabling real-time situational awareness and cyber-physical control through a digital twin for traffic, IEEE Trans. Intell. Transp. Syst. (2023).
[60] J. Pereira, T. Batista, E. Cavalcante, A. Souza, F. Lopes, N. Cacho, A platform for integrating heterogeneous data and developing smart city applications, Future Gener. Comput. Syst. (ISSN: 0167-739X) 128 (2022) 552–566, http://dx.doi.org/10.1016/j.future.2021.10.030, URL https://www.sciencedirect.com/science/article/pii/S0167739X2100426X.
[61] A. Zanella, N. Bui, A. Castellani, L. Vangelista, M. Zorzi, Internet of things for smart cities, IEEE Internet Things J. 1 (1) (2014) 22–32.
[62] S. Sengan, V. Subramaniyaswamy, S.K. Nair, V. Indragandhi, J. Manikandan, L. Ravi, Enhancing cyber–physical systems with hybrid smart city cyber security architecture for secure public data-smart network, Future Gener. Comput. Syst. 112 (2020) 724–737.
[63] A. Abbasi, A.R. Javed, C. Chakraborty, J. Nebhen, W. Zehra, Z. Jalil, ElStream: An ensemble learning approach for concept drift detection in dynamic social big data stream learning, IEEE Access 9 (2021) 66408–66419.
[64] B. Krawczyk, A. Cano, Online ensemble learning with abstaining classifiers for drifting and noisy data streams, Appl. Soft Comput. 68 (2018) 677–692.
[65] Y. Sun, Z. Wang, Y. Bai, H. Dai, S. Nahavandi, A classifier graph based recurring concept detection and prediction approach, Comput. Intell. Neurosci. 2018 (2018).
[66] R. Mohawesh, S. Tran, R. Ollington, S. Xu, Analysis of concept drift in fake reviews detection, Expert Syst. Appl. 169 (2021) 114318.
[67] A.S. Iwashita, J.P. Papa, An overview on concept drift learning, IEEE Access 7 (2018) 1532–1547.
[68] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: Brazilian Symposium on Artificial Intelligence, Springer, 2004, pp. 286–295.
[69] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. (CSUR) 46 (4) (2014) 1–37.
[70] I. Frias-Blanco, J. del Campo-Ávila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-Diaz, Y. Caballero-Mota, Online and non-parametric drift detection methods based on Hoeffding's bounds, IEEE Trans. Knowl. Data Eng. 27 (3) (2014) 810–823.
[71] G. Schreier, P.M. Prank, Fault tolerant ship propulsion control: sensor fault detection using a nonlinear observer, in: 1999 European Control Conference, ECC, IEEE, 1999, pp. 4586–4591.
[72] P. Mulinka, P. Casas, J. Vanerio, Continuous and adaptive learning over big streaming data for network security, in: 2019 IEEE 8th International Conference on Cloud Networking, CloudNet, IEEE, 2019, pp. 1–4.
[73] O.A. Mahdi, E. Pardede, N. Ali, J. Cao, Diversity measure as a new drift detection method in data streaming, Knowl.-Based Syst. 191 (2020) 105227.
[74] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, R. Morales-Bueno, Early drift detection method, in: Fourth International Workshop on Knowledge Discovery from Data Streams, Vol. 6, 2006, pp. 77–86.
[75] R.S. Barros, D.R. Cabral, P.M. Gonçalves Jr., S.G. Santos, RDDM: Reactive drift detection method, Expert Syst. Appl. 90 (2017) 344–355.
[76] G.J. Ross, N.M. Adams, D.K. Tasoulis, D.J. Hand, Exponentially weighted moving average charts for detecting concept drift, Pattern Recognit. Lett. 33 (2) (2012) 191–198.
[77] A. Pesaranghader, H.L. Viktor, Fast hoeffding drift detection method for evolving data streams, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2016, pp. 96–111.
[78] B. Milosevic, E. Farella, Wireless MEMS for wearable sensor networks, in: Wireless MEMS Networks and Applications, Elsevier, 2017, pp. 101–127.
[79] PROJECT – BRAINE. URL https://www.braine-project.eu/project/.
[80] Apache Hadoop Ozone. URL https://hadoop.apache.org/ozone/.
[81] N. Alange, A. Mathur, Optimization of small sized file access efficiency in hadoop distributed file system by integrating virtual file system layer, Optimization 13 (6) (2022).
[82] R. Ali, Y.A. Qadri, Y.B. Zikria, F. Al-Turjman, B.-S. Kim, S.W. Kim, A blockchain model for trustworthiness in the internet of things (IoT)-based smart-cities, Trends Cloud-Based IoT (2020) 1–19.
[83] Welcome to Apache Solr. URL https://solr.apache.org/index.html.
[84] Apache Spark™ - Unified Engine for large-scale data analytics. URL https://spark.apache.org/.
[85] Apache Storm. URL https://storm.apache.org/.
[86] Fixing the analytics models that COVID-19 broke | McKinsey, 2022, https://www.mckinsey.com/capabilities/quantumblack/our-insights/leaderships-role-in-fixing-the-analytics-models-that-covid-19-broke. (Online; Accessed 13 October 2022).
[87] C. Le Quéré, R.B. Jackson, M.W. Jones, A.J. Smith, S. Abernethy, R.M. Andrew, A.J. De-Gol, D.R. Willis, Y. Shan, J.G. Canadell, et al., Temporary reduction in daily global CO 2 emissions during the COVID-19 forced confinement, Nat. Clim. Change 10 (7) (2020) 647–653.
[88] D. Sykes, Domestic energy usage patterns during social distancing, 2020, URL https://octopus.energy/blog/domestic-energy-usage-patterns-during-social-distancing/.
[89] G. Nicolini, G. Antoniella, F. Carotenuto, A. Christen, P. Ciais, C. Feigenwinter, B. Gioli, S. Stagakis, E. Velasco, R. Vogt, et al., Direct observations of CO2 emission reductions due to COVID-19 lockdown across European urban districts, Sci. Total Environ. 830 (2022) 154662.
[90] H.M. Gomes, J. Montiel, S.M. Mastelini, B. Pfahringer, A. Bifet, On ensemble techniques for data stream regression, in: 2020 International Joint Conference on Neural Networks, IJCNN, IEEE, 2020, pp. 1–8.
[91] J. Schmidhuber, S. Hochreiter, et al., Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[92] D. Xia, M. Zhang, X. Yan, Y. Bai, Y. Zheng, Y. Li, H. Li, A distributed WND-LSTM model on MapReduce for short-term traffic flow prediction, Neural Comput. Appl. 33 (7) (2021) 2393–2410.
[93] T. Ergen, S.S. Kozat, Online training of LSTM networks in distributed systems for variable length data sequences, IEEE Trans. Neural Netw. Learn. Syst. 29 (10) (2017) 5159–5165.
[94] P. Filonov, A. Lavrentyev, A. Vorontsov, Multivariate industrial time series with cyber-attack simulation: Fault detection using an lstm-based predictive data model, 2016, arXiv preprint arXiv:1612.06676.
[95] N. Jrad, A. Kachenoura, A. Nica, I. Merlet, F. Wendling, A Page-Hinkley based method for HFOs detection in epileptic depth-EEG, in: 2017 25th European Signal Processing Conference, EUSIPCO, IEEE, 2017, pp. 1295–1299.
[96] C. Raab, M. Heusinger, F.-M. Schleif, Reactive soft prototype computing for concept drift streams, Neurocomputing 416 (2020) 340–351.
[97] Tellus | University of Oulu. URL https://www.oulu.fi/en/cooperation/tellus.
[98] Carbon Dioxide, Wisconsin Department of Health Services, 2018, URL https://www.dhs.wisconsin.gov/chemical/carbondioxide.htm.
[99] A.S. Iwashita, V.H.C. de Albuquerque, J.P. Papa, Learning concept drift with ensembles of optimum-path forest-based classifiers, Future Gener. Comput. Syst. 95 (2019) 198–211.
[100] E.B. Gulcan, F. Can, Unsupervised concept drift detection for multi-label data streams, Artif. Intell. Rev. 56 (3) (2023) 2401–2434.
[101] M. Karimian, H. Beigy, Concept drift handling: A domain adaptation perspective, Expert Syst. Appl. 224 (2023) 119946.
[102] Apache Ranger – Introduction. URL https://ranger.apache.org/.
[103] Apache Atlas – Data Governance and Metadata framework for Hadoop. URL https://atlas.apache.org/#.



Hassan Mehmood is a doctoral researcher at the Center for Ubiquitous Computing, University of Oulu, Finland. His research interests include distributed computing, data management and integration, machine learning, big data analytics, edge computing, and concept drift detection and adaptation in real-time data streams. Contact him at hassan.mehmood@oulu.fi.



Ahmed Khalid is a Senior Research Scientist with Dell Technologies based in the Centre of Excellence, Cork, Ireland. His research interests also include network and service orchestration, IaaS, cloud and edge computing, cryptography, E2E security, intelligent data management, automated deployment, and resource management. Contact him at ahmed.khalid@dell.com.

Panos Kostakos is a Senior Research Fellow at the Center for Ubiquitous Computing, University of Oulu, Finland. His research is situated at the intersection of AI, Information Security, and Security orchestration, focusing on autonomous, mutable, and cognitive cyber defence mechanisms. He leads the research group Cyber Security Informatics (CSI). Contact him at panos.kostakos@oulu.fi.

Ekaterina Gilman is a postdoctoral researcher supported by the Academy of Finland at the Center for Ubiquitous Computing, University of Oulu, Finland. Her research interests include data analytics, context modelling and reasoning, and machine learning in ubiquitous computing, IoT, and data-intensive systems. Contact her at ekaterina.gilman@oulu.fi.

Susanna Pirttikangas is deputy director for the Center for Ubiquitous Computing at the University of Oulu and the principal investigator of the Interactive Edge (iEdge) research team developing adaptive, reliable, and trusted edge intelligence. Research director Pirttikangas has an extensive background in artificial intelligence-related research and business. Contact her at susanna.pirttikangas@oulu.fi.