Journal of Cloud Computing

RESEARCH Open Access

A framework for the predictive monitoring and data quality assurance in the cloud continuum

Lander Bonilla[1*], Josu Diaz-de-Arcaya[1], Juan López-de-Armentia[1] and Aitor Almeida[2]

Abstract
The widespread adoption of the cloud continuum paradigm poses significant challenges such as the management of heterogeneous infrastructural devices, the strict security and privacy requirements, and the complex data governance constraints. While the cloud provides access to advanced services that are often inaccessible to small and medium organizations, edge and fog resources are essential to meet latency, locality, and efficiency demands. The main benefits provided by the cloud continuum is to provide scalability, flexibility, and resilience by seamlessly integrating networking, storage, and computing resources across different layers. In this context, we present a lightweight agent, coined EdgeGuard, designed for seamless integration into heterogeneous infrastructures within a computing continuum architecture. It enables real-time monitoring of multiple metrics (e.g., resource utilization, energy consumption) and provides predictive capabilities to anticipate and mitigate potential issues before they escalate. We validate our proposal through an experimental scenario involving a diverse set of infrastructural devices distributed across the continuum with experts in the field. The evaluation shows that EdgeGuard consistently outperforms human experts across all measured metrics. These results highlight its effectiveness in proactive monitoring and correction of infrastructural issues, making it a suitable tool for modern distributed computing environments. Ultimately, EdgeGuard contributes to building more resilient, scalable, and intelligent systems within the evolving landscape of edge-cloud continuum.

Keywords Cloud continuum, AIIoT, Distributed monitoring, Predictive maintenance, Hybrid cloud-edge architecture, AIOps

Introduction

The convergence of artificial intelligence and the cloud continuum is at the forefront of a technological revolution that has the potential to completely transform our digital landscape [1]. In recent years, the exponential growth of data generation, combined with an insatiable demand for real-time insights, has highlighted the critical need for innovative solutions capable of processing and analyzing data across heterogeneous and distributed infrastructures [2, 3]. The cloud continuum, with its seamless integration of cloud, fog, and edge resources, has emerged as a critical paradigm to meet

*Correspondence:
Lander Bonilla
lander.bonilla@tecnalia.com
[1]Tecnalia, Basque Research & Technology Alliance (BRTA), Albert Einstein 11, Miñano, Álava 01510, Spain
[2]DeustoTech, University of Deusto, Avenida de las Universidades 48007, Bilbao, Biscay 48007, Spain

these demands by dynamically allocating computational resources across the network hierarchy [4]. However, the growing volume of data generated at the edge has rendered traditional cloud-centric paradigms less efficient [5]. This shift has catalyzed the rise of Edge AI, which promotes lowering AI models down to the edge layer to leverage benefits such as speed, localized security, and reliability [6, 7]. Concurrently, AI algorithms, with their ability to extract actionable intelligence from large datasets, have transformed decision-making processes across industries. AIIoT and cloud continuum work in tandem to enhance the agility and scalability of distributed infrastructures with the intelligence and automation capabilities of algorithms [8, 9]. As these environments expand, the use of standardizing management and monitoring tools becomes vital to simplify deployment and minimize human error [10].

This convergence has far-reaching implications across multiple domains, but it also implies a series of challenges that must be considered when deploying cloud continuum environments. One such challenge is the current lack of adequate monitoring elements capable of covering the edge while maintaining resilience and non-intrusiveness [11]. Effective oversight requires a multi-layered approach, encompassing VM-level, container-level, link quality, and application-level monitoring to optimize behavior and responsiveness at the edge [12]. One challenge highlighted in research is the issue of security and privacy in distributed infrastructures [13, 14]. Documented vulnerabilities in AI models and device access necessitate robust intrusion detection and certificate management [15, 16]. As data flows and is processed across multiple layers of the continuum, new vulnerabilities and threats arise. Another critical challenge is optimizing resource orchestration, given the heterogeneity of devices and the varying computing capacities of continuum layers [17, 18]. Balancing cost, latency, and performance trade-offs is crucial for ensuring efficient resource allocation and utilization in cloud continuum systems.

Furthermore, the application of AIOps techniques is increasingly necessary to ensure the quality and integrity of data processed at the edge [19]. Since operational decisions rely on this data, monitoring must address anomalies and maintain consistency as close as possible to the point of collection [20, 21]. Moreover, the scalability of the cloud continuum poses a significant challenge, especially when dealing with a growing number of IoT devices generating massive amounts of data and requesting computation offloading [22, 23]. Developing frameworks and architectures that can scale effectively to accommodate increasing demands is essential for the widespread adoption of the cloud continuum paradigm.

Problem statement

In the realm of the cloud continuum and artificial intelligence, edge devices have emerged as critical elements for real-time data collection, processing, and transmission at the network's edge [24]. These AI-driven IoT devices perform several processes and applications that are critical to the efficient operation of distributed systems [25]. However, effectively monitoring these processes on edge devices presents several challenges that require careful consideration.

Firstly, there is a diverse range of platforms and architectures upon which edge devices are built, making it challenging to implement universally applicable monitoring solutions. Secondly, many edge devices have limited computational, memory, and energy resources, imposing significant constraints on implementing monitoring systems that consume additional resources. Moreover, these devices may experience intermittent connectivity or operate in low-quality network conditions, complicating the reliable real-time transmission of monitoring data. As the number of edge devices increases in IoT and cloud computing deployments, monitoring must be scalable and capable of managing the large number of geographically distributed devices. Furthermore, ensuring the security and privacy of the collected monitoring data, especially when dealing with sensitive data. Lastly, effectively identifying and responding to anomalies in processes executed on edge devices is essential for ensuring system integrity and operability. Addressing these challenges is imperative to develop resource-efficient, interoperable, secure, and dynamically operable process monitoring solutions for edge devices. Doing so will enhance the management, performance, and security of edge device-based systems, paving the way for broader and more effective adoption of edge computing technologies.

Contributions

In this paper we present a framework, coined *EdgeGuard*, that focuses on monitoring and ensuring the efficiency of heterogeneous environments. The main contributions of this work can hence be summarized as follows:

C1 A real-time, multi-level monitoring methodology is presented, which facilitates customization of monitoring on each device while also providing scalability to the solution.
C2 The formulation for data quality measurement at the network edge is explained, achieving real-time validation of data before processing and facilitating the integration of quality standards into devices.
C3 A framework that can be seamlessly deployed across cloud continuum environments, supporting

the efficient management of distributed systems and ensuring effective resource utilization.
C4 A metric forecasting system combining predictive algorithms with real-time monitoring, which promotes predictive maintenance, resource optimization, and well informed decision-making.
C5 An abstract architecture that unifies monitoring and self-repair capabilities across the cloud continuum, enabling multi-layered management of heterogeneous and dynamic environments.

The rest of the paper is organized as follows. In Section "Related work", scientific studies in the field of predictive monitoring are analyzed. Next, the architecture of the system is detailed in Section "System overview". Section "Experimental results" elaborates on various research questions associated with the contributions of this manuscript. An expert evaluation is showcased in Section "Expert evaluation". Finally, the results are discussed in Section "Discussion" followed by the conclusions and future work (Section "Conclusions and future work").

Related work

Monitoring at the network edge is one of the key points for the proper functioning of edge environments within industry. There is a wide variety of tools that perform this task at the edge. In Table 1, we offer a comparison of various technologies, evaluated under these features.

F1 Real time monitoring: the agent supports the monitoring of devices in real time.

F2 Forecast monitoring: the agent is capable of predicting the state of the devices to take appropriate actions.

F3 Data Quality: it supports the monitoring of the quality of the IoT data.

F4 Multi-Level monitoring: it can monitor various levels within devices, including machine level, container level, network level, data level, and application level.

F5 Multi-layer: it can be deployed across cloud continuum environment layers (Edge, Fog, Cloud).

F6 Scalability: it can function optimally increasing its performance in accordance with the demand.

F7 Open-source: it is freely available for anyone to inspect, modify, and distribute, promoting collaboration and community driven development.

F8 Technology agnostic: the solution is designed to work with a variety of technologies.

These features have been selected from the scientific literature to address significant challenges in our research. In [42] the authors outline a series of issues that an edge computing environment must be capable of addressing. Among these challenges is the *Service Management* challenge, which discusses the difficulty that edge faces in controlling the various services running on it. That's why we have decided that both the current monitoring (F1) and the forecast monitoring (F2) are important. Moreover, the different monitoring levels (F4) mentioned in

Table 1 Tools that support the deployment of machine learning projects in production environments

	Real time monitoring	Future monitoring	Data Quality	Multi-Level monitoring	Multi-layer	Scalability	Open-source	Technology Agnostic
Monit [26]	✓	×	×	×	✓	✓	✓	✓
PyMon [27]	✓	×	×	×	✓	✓	✓	✓
Prometheus Stack [28]	≈	×	×	×	✓	✓	✓	✓
Zenoss [29]	✓	×	≈	×	✓	✓	✓	✓
Ganglia [30]	✓	×	×	≈	×	✓	✓	×
Zabbix [31]	✓	×	×	×	≈	✓	✓	×
Nagios [32]	✓	×	×	✓	✓	×	✓	✓
OpenNebula [33]	✓	×	✓	≈	✓	≈	✓	✓
PCMONS [34]	✓	×	×	×	≈	×	✓	×
DARGOS [35]	✓	×	×	×	≈	×	✓	✓
Lattice [36]	✓	×	×	×	×	×	✓	✓
JCatascopia [37]	✓	×	×	✓	≈	≈	✓	×
Tower 4Clouds [38]	≈	×	×	×	×	×	✓	✓
Talend [39]	×	×	✓	×	✓	≈	×	×
OpenRefine [40]	×	×	✓	×	✓	≈	✓	≈
IBM InfoSphere [41]	×	×	✓	×	✓	≈	×	×
EdgeGuard	✓	✓	✓	✓	✓	✓	✓	✓

[12] are considering adding a new one. Also, the measurement of Data Quality (F3) of the data that the devices are working with add value to the agent. In addition, edge computing studies [43] emphasize the scalability (F6) of solutions and deployment across multiple layers (F5). The reason for positively valuing being open-source (F7) project is because they are easy to use and build upon [44]. Besides, we evaluate technology agnostic (F8) projects because they offer flexibility, future-proofing, and interoperability across diverse technologies [45].

Being able to monitor the status of edge computing is becoming the key for the success of different projects, that's why there are different tools that do a similar job to ours. In this line, Monit [26] is a monitoring tool designed for system monitoring and management, overseeing processes, resources, and files, and taking automatic actions upon issues like process failures or resource constraints. Monit may consume system resources and lacks extensive capabilities for complex or distributed monitoring compared to other tools. Configuring Monit requires specific knowledge and accurate settings to avoid misconfigurations that could impact monitoring effectiveness. In addition, PyMon [27] is a tool that utilizes Monit behind the scenes and adds the capability to monitor containers running on the system and it showed lower CPU overhead compared to the other solutions, making it suitable edge enviroments.

Prometheus Stack [28] is an open-source monitoring and alerting toolkit designed for reliability and scalability of systems and applications. It collects metrics from configured targets, stores them locally, and enables querying and visualization of these metrics in real-time. Zenoss [29] is an open-source monitoring tool that focuses on network, server, and application monitoring. It offers scalability, robustness, and interoperability. However, it may have limitations in adaptability to dynamic environments like edge computing. However, its lack of long-term storage capabilities, so it requires the integration with other systems for persistent data. Ganglia [30] is a distributed monitoring tool designed for high-performance computing environments such as grids and clusters. It uses technologies like Extended Detection and Response (XDR), Extensible Markup Language (XML), and Round Robin Database Tool (RRDtool) for storing and visualizing time series monitoring data. While scalable, Ganglia may not be as suitable for edge computing due to limitations in bulk data transfer capabilities.

Zabbix [31] is a monitoring tool that provides monitoring for networks, servers, and applications. It features a server/agent architecture and supports an alerting system. It offers essential monitoring capabilities but may have limitations in dynamic environments like edge. Nagios [32] is an open-source solution for monitoring network and infrastructure resources. While widely used, it may not be as suitable for dynamic environments like edge due to its design and capabilities.

There are also different cloud monitoring solutions that could work in edge environments. For example, OpenNebula [33] is a cloud management platform with monitoring capabilities. It also provides monitoring solutions for edge environments but may have limitations in terms of scalability and adaptability to different environments. PCMONS [34] is a monitoring system designed for private clouds, it offers monitoring capabilities for private cloud infrastructures. However, it may have limitations in terms of scalability and adaptability for dynamic edge environments. DARGOS [35] is a decentralized resource monitoring solution for cloud computing infrastructures that aims to provide monitoring capabilities for cloud environments, but may have the same limitations that PCMONS has with the scalability.

Lattice [36], from the Reservoir [46] project, is a monitoring framework for federated clouds that offers monitoring solutions for federated cloud environments but may have limitations in terms of CPU consumption. JCatascopia [37] is a cloud monitoring tool that provides features for monitoring applications deployed on edge computing frameworks. It offers functionalities to meet various monitoring requirements, but like other tools, it may not fully address all the needs of monitoring in edge computing environments. JCatascopia has been designed to ensure the performance and reliability of applications, but it may have limitations in terms of adaptability and comprehensive monitoring capabilities. Tower 4Clouds [38] is a multi-cloud monitoring platform developed as part of the MODAClouds [38] project. It collects metrics at VM and application levels through self-registering data collectors and stores the monitoring data in InfluxDB [47] or Graphite [48]. The platform continuously checks a set of monitoring rules to determine application health and QoS constraints, triggering actions when rules become true. The problem lies in the fact that it is not a tool designed for edge environments.

If we move from monitoring the state of devices to monitoring data quality, we find the following tools. Talend [39] is a data quality monitoring solution designed to enhance data management processes. It leverages machine learning technology to profile, clean, and mask data in real-time, providing intelligent recommendations to address data quality issues. Its machine learning-enabled deduplication, validation, and standardization functions allow users to clean incoming records and enrich them as needed, ensuring access to reliable information. However, setting up Talend Data Quality can be complex, particularly for non-technical users. The solution also lacks in-memory capability, leading to potential performance and speed issues, especially with large and complex data transformation tasks. Additionally, it

comes at a higher price compared to several other Data Quality Management solutions in market.

OpenRefine [40] is an open-source tool for data quality management. With this tool, you can identify and correct data issues, apply data transformations, and perform data explorations. It offers a variety of features for cleaning and standardizing data to ensure accuracy and consistency. However, the tool comes with certain limitations, since reusing workflows can be challenging due to the lack of error handling in operation sequences and the adaptation of workflows to projects with different column names. IBM InfoSphere Information Server [41] is a data integration platform that simplifies understanding, cleaning, tracking, and transforming data. It enables continuous data cleansing and tracking, allowing organizations to convert raw data into reliable information. But it includes a complex initial setup that requires technical expertise, and smaller or less complex entities may find it excessive for their needs.

△=△In summary, as illustrated in Table 1, there is still a big need for a comprehensive solution that is especially suited to the subtleties of the edge, even though the examined landscape provides strong capabilities for certain monitoring tasks. Established monitoring tools excel in real-time metrics and infrastructure stability, but they remain primarily reactive and restricted in functionality, failing to address the critical need for predictive maintenance (F2) in edge environments. Furthermore, specialized data quality platforms are often too resource intensive, complex to configure, or prohibitively licensed for deployment within the resource-constrained layers of the cloud continuum. EdgeGuard effectively addresses these shortcomings, surpassing the state of the art by providing a unified framework that balances high scalability with advanced features like Data Quality (F3) and Multi-Level monitoring (F4). Unlike its predecessors, which often prioritize one capability at the expense of others, EdgeGuard integrates proactive forecasting with a lightweight, multi-layer architecture designed for the entire cloud-to-edge spectrum. By anticipating failures through predictive modeling while simultaneously ensuring the integrity of IoT data streams, it provides the comprehensive oversight and technology-agnostic flexibility necessary for the success of complex projects.

System overview

In this section, the model architecture where EdgeGuard resides is explained in Section "Architecture overview". We also offer an in-depth explanation of the main elements and stages of the EdgeGuard flow, that is described in Section "EdgeGuard workflow". In Section "Agent data schemas", all the data managed by the agent is shown. Afterwards, in Section "Prediction and status check", using this collected data, the agent predicts the future and decides whether to launch alerts. In Section "Data quality management", the management of data quality in streaming data is explained, and lastly, in Section "Multi-layer deployment", explains the ability of the agent to be deployed in various computational layers. Python 3 was used to implement the agent because it is cross-platform and comes pre-packaged in many different operating systems.

Architecture overview

The system architecture, presented in Fig. 1, is centered around two synergistic components, the Monitoring Agent (EdgeGuard) and the Agentic Recovery, which are deployed throughout the cloud continuum, encompassing the Edge, Fog, and Cloud computational layers. Rather than treating these layers as isolated domains, it leverages their interconnection to enable a seamless, intelligent, and autonomous self-healing system. This design ensures that failures are detected and resolved at the most appropriate level, whether locally at the edge or

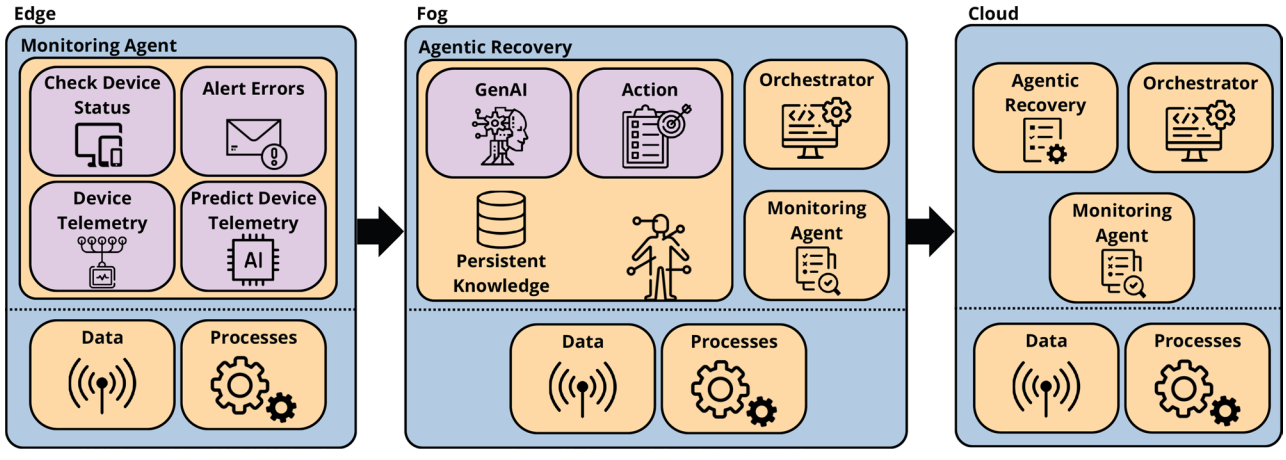

Fig. 1 Architectural diagram of EdgeGuard integration with predictive monitoring and self-healing capabilities

globally in the cloud, depending on the nature, scope of the issue and the required amendment.

EdgeGuard is in charge of continuously gathering and analyzing telemetry data from all layers of devices and services. It functions in close proximity to the physical devices at the edge, allowing for real-time monitoring of operational metrics, environmental conditions, and hardware status. Because of this close proximity, anomalies are detected during the earlier stages, and problems are promptly addressed. Broader situational awareness is made possible by the Monitoring Agent's aggregation and contextualization of data as it progresses through the upstream layers. It correlates information from several edge nodes in the fog layer to find regional trends or cascade failures. It performs large-scale analytics in the cloud, using AI models and historical data to forecast possible system-wide outages. This multi-scale monitoring capability ensures that the system remains aware of both micro-level device behavior and macro-level infrastructure trends.

The *Agentic Recovery* component represents the system's independent decision-making and self-healing services. A dynamic and changing body of *Persistent Knowledge*, a storehouse of previous episodes, recovery acts, and learning strategies, drives the *Agentic Recovery* rather than a static rule-based system. It assesses the situation, looks through its knowledge base, and triggers the most appropriate recovery action when the EdgeGuard notices an irregularity or anticipates a failure. This could entail the isolation of a problematic component, reallocating resources, or restarting a service at the edge. It may coordinate recovery across several nodes in the fog layer, guaranteeing service continuity even in the event of partial failures, while maintaining adherence to service-level agreements. It coordinates extensive interventions including traffic rerouting, patch deployment, and service reconfiguration at the cloud layer level.

EdgeGuard workflow

A sequence of steps that must be completed for EdgeGuard to yield and an acceptable deployment definition are shown in detail in Fig. 2. First, in **step 1**, a monitoring of the status of IoT devices is done, which involves collecting a variety of data points essential for understanding the device's state and functioning. The metrics that have been considered are explained in Section "Agent data schemas". This continuous data collection is foundational for both real-time monitoring and future predictive analysis. In parallel, in this same step, the data utilized by the devices at the edge is processed. This data consists of raw data collected from different sensors connected to the devices, and it is essential to ensure its quality for optimal use, both for AI at the edge and in the cloud. According to DAMA [49], a check is performed on the six necessary dimensions of data quality, explained in more detail in Section "Data quality management".

Next, in **step 2**, with the data collected in the previous step, a predictive analysis is performed leveraging machine learning algorithms to forecast potential problems. By identifying patterns and correlations between the different metrics, the system can anticipate future problems based on current data. This step enables predictive maintenance, allowing issues to be addressed before they result in device malfunctions. This forward-looking approach helps minimize downtime and extend the operational life of the device. Section "Prediction and status check" explains in more depth the prediction that is carried out. Once we have the current data and the device's predictions, they are checked to see if they fall within normal limits. This way, we can determine if the device has or will have any issues, this is done in **step 3**.

When issues are detected in data processed in **step 1** and **2**, the system will create an alert for the next computational layer, **step 4**. The alert includes detailed information about the device, the issue of the device, and the recommended action that can be taken to resolve it, in Section "Prediction and status check" few alerts can be seen along with their respective proposals. This notification provides the next computational layer with the information to solve the problem, ensuring that complex problems are addressed and resolved promptly. This step ensures that all device issues, whether simple or complex, are handled effectively, maintaining the overall health and functionality of the environment.

Agent data schemas

In this section, the different data schemas used by the monitoring agent will be introduced. Taherizadeh et al. [12] describe four types of edge monitoring in their article. The development of the agent has focused on these types of monitoring, adding to this the monitoring

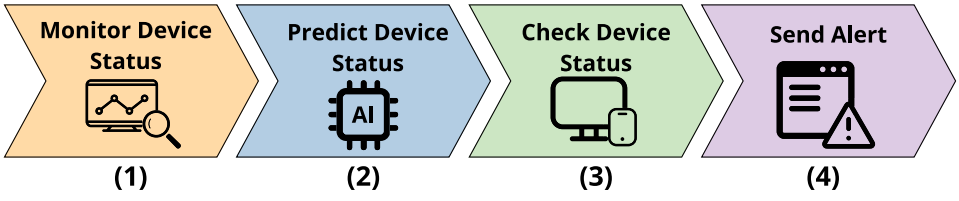

Fig. 2 Agent workflow for monitoring and prediction in IoT devices

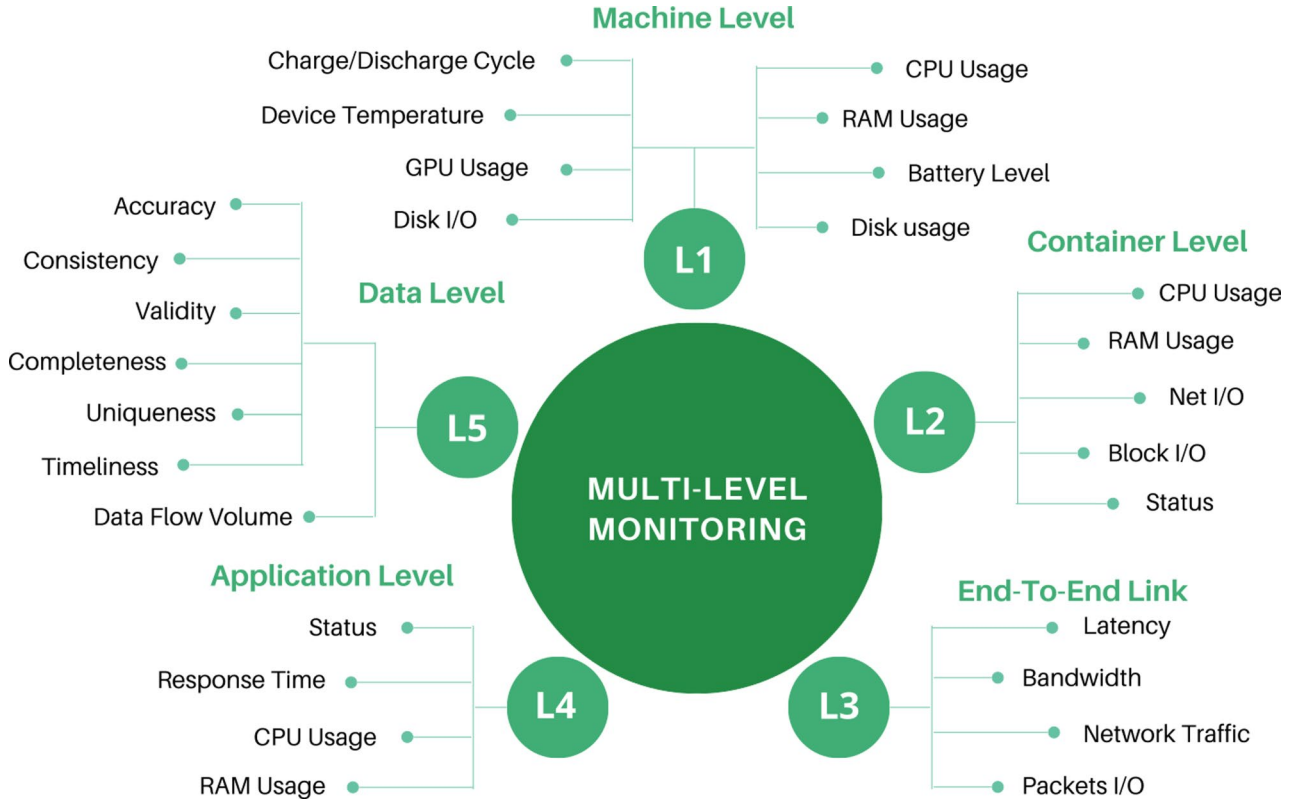

Fig. 3 Monitoring levels for comprehensive device and system performance assessment

of the device data. Figure 3 shows the different levels of monitoring:

L1 Machine-level monitoring This involves the detailed supervision of computing resources on individual devices, providing critical insights into their performance. This process includes observing and analyzing key metrics such as CPU usage, memory consumption, disk space, and battery usage. By closely monitoring these metrics, it ensures that devices operate optimally and efficiently, helping to identify potential issues before they impact system performance. Additionally, it allows for proactive resource management, improving the longevity and reliability of individual machines in a network. To measure these aspects, metrics such as CPU usage, RAM usage, battery level, disk usage, device temperature, charge/discharge cycle, GPU usage, disk I/O, fan speed (RPM), and voltage have been used.

L2 Container-level monitoring This type of monitoring focuses specifically on Docker containers and involves tracking metrics unique to containerized applications, such as resource usage (CPU, memory, disk I/O and block I/O), runtime values of key attributes, and performance indicators. This type of monitoring is crucial for managing and optimizing containerized workloads in both cloud and edge computing environments, ensuring containers run efficiently and without resource contention. It also helps in detecting issues like resource leaks, inefficient scaling, or performance bottlenecks, which can compromise application stability and performance. To measure these, metrics such as CPU usage, RAM usage, net I/O, block I/O, container status, and image count have been utilized.

L3 End-to-end link monitoring This type of monitoring is concerned with measuring the quality of network connections between different application components to ensure consistent and reliable communication across the system. It evaluates various end-to-end network Quality of Service (QoS) properties, such as latency, bandwidth or packet loss, to assess the overall health and performance of the communication links. Additionally, it supports on-demand network configuration and optimization to maintain or improve link quality, helping to avoid disruptions and ensure smooth data flow across distributed systems. To measure these properties, metrics such as latency, bandwidth, network traffic, packets I/O, packet loss rate, and signal-to-noise ratio have been used.

L4 Application-level monitoring It involves measuring metrics directly related to the performance and reliability of an application, such as service response time, error rates, application status, and hardware usage. This type

of monitoring is crucial for assessing the overall health and efficiency of cloud-based applications, ensuring they meet performance expectations and provide a positive user experience. By continuously tracking these metrics, it helps identify bottlenecks, optimize resource usage, and maintain application's responsiveness and availability. To measure these, metrics such as application status, response time, CPU usage, RAM usage, and error rates have been used.

L5 Data-level monitoring Data monitoring is a crucial aspect of data management that involves continuously observing, reviewing, and analyzing data to maintain its quality, integrity, and performance. This process ensures that data remains accurate, reliable, and free from inconsistencies or errors, which is vital for effective decision-making. Data monitoring also helps ensure compliance with regulatory standards, minimizing risks associated with data breaches or inaccuracies. To measure these, metrics such as accuracy, consistency, validity, completeness, uniqueness, timeliness, and data flow volume have been used.

Within each of these monitoring levels, Table 2 shows the different alerts that have been created for each level, what is the severity of the error and what is the possible solution that the agent will execute. To ensure that the agent monitors the desired parameters, its deployment is done through a configuration file. This file specifies various values for the agent, such as where it is deployed, the type of monitoring to be performed, which processes are important on the device, and the criteria for detecting errors in different metrics. Crucially, all metrics and their corresponding alert thresholds are fully configurable, allowing the system to be tailored to the specific operational requirements and tolerance levels of different environments.

Prediction and status check

Once both the base configuration of the agent and the different monitoring levels at which it operates have been defined in Subsection "Agent data schemas", it is necessary to explain how the agent collects data, predicts how that data will be in future, and from there, draws conclusions.

To carry out this prediction at the edge, the Python machine learning library called River ML has been used [50]. River is an online machine learning model building library. These models work with streams of data. A data

Table 2 Categories of agent alerts for effective performance monitoring and issue detection

Level	Name	Description	Severity	Recommended Action
L1	Low Battery Level	Battery level below a threshold	High	Connect to power source, check battery health, consider replacing battery
	High CPU Usage	CPU usage exceeds threshold	High	Investigate running processes, optimize or upgrade hardware
	High RAM Usage	RAM usage exceeds threshold	High	Investigate running processes and check for memory leaks
	Little Disk Space	Disk space below a threshold	Medium	Clean up unnecessary files, consider expanding storage
	Disk Error	Disk read/write error detected	High	Check disk health status, run disk repair tools, consider replacing the disk
	High Device Temperature	Device temperature exceeds threshold	High	Check cooling system files and change the minimum temperature
L2	High Container CPU Usage	Container CPU usage exceeds threshold	High	Limit the CPU usage and restart container
	High Container RAM Usage	Container RAM usage exceeds threshold	High	Limit the memory usage and restart container
	Little Container Disk Space	Container disk below a threshold	Medium	Give more disk space to the container
L3	No External Connection	Device cannot connect to external networks or services	High	Restart network settings
	Overloaded Network	Network latency exceeds threshold	Medium	Restart connections and the faulty processes
	No Broker Connection	Connection to the broker returns failed	High	Restart the broker if possible or send an external alert
L4	High App Response Time	Application response time exceeds acceptable threshold	Medium	Check app performance or restart it
	Application Crash	Critical application has crashed	High	Restart the application, analyze crash reports
	QoS Degradation	Quality of Service metrics below acceptable levels	High	Restart network and application
L5	No Data Stream	One or more connected sensors are not reporting data	Medium	Check sensor connections, replace faulty sensors
	Changes in Data Flow Volume	Significant increase or decrease in data flow volume	Low	Check for bottlenecks or data loss issues
	Data Quality Problem	Repeated data quality issues	Medium	Check data sources

stream is often an ordered collection of discrete components. Each component in machine learning instance is made up of several features. These are what we refer to as observations. Every sample may adhere to a set format and consistently include the same elements. However, characteristics can also change over time [50]. The architecture aims to facilitate the deployment of stream learning in several areas, including industrial applications and academic research, by being both flexible and user-friendly [51].

Algorithm 1 is designed to continuously monitor a device by leveraging three main inputs: a configuration file, a River machine learning model, and the threshold values. It begins by initializing these components and then enters a monitoring loop that persists until an alert condition is met. Within this loop, the algorithm systematically processes each monitoring level defined in the configuration file.

For each monitoring level, the algorithm first calculates a future time based on the settings in the file. This future time frame, known as the horizon, is crucial for forecasting the state of the device at a certain moment in time. Next, it extracts the monitored data (denoted as **m**) relevant to the current monitoring level. This data represents the historical and current operational parameters of the device, which are crucial for training the machine learning model. Once the data is retrieved, the algorithm updates the model incrementally, allowing it to learn from the new data while retaining knowledge from previous data. After the model is trained, a prediction of the device's status at the calculated future time is made, based on the trained model and the monitored data, achieving a real-time prediction.

Algorithm 1 Device Status Prediction

Require: conf.yaml, model, threshold

while $True$ **do**
  for $L_i \in conf.yaml$ **do** ▷ $L_i = MonitoringLevel$
    $md \leftarrow monitoredData \in L_i$
    $model \leftarrow model.learnOne(md)$
    $predictData \leftarrow model.forecast(horizon)$
    if $predictData \notin threshold$ **then**
      $alert \leftarrow Status + Action$
      $sendAlert(error) \leftarrow predictData$
    end if
  end for
end while

The core of the monitoring process lies in comparison of the predicted data against the predefined thresholds. If the predicted value exceeds the threshold, it indicates a potential issue with the device. Consequently, an alert is triggered, and the algorithm captures the error value associated with this prediction. This alert is sent with the predicted values and the information of the device, so an external solution can be applied.

By continuously training the model and predicting the device's future state, the algorithm provides a proactive approach to device maintenance. It can anticipate problems before they occur, allowing for timely interventions and minimizing downtime. The integration of machine learning ensures that the algorithm adapts to changing patterns in device's operation, enhancing its predictive accuracy and reliability over time.

Data quality management

Data quality in Internet of Things (IoT) is crucial for ensuring the reliability, accuracy, and usability of data collected from various devices. High-quality IoT data is essential for informed decision-making, efficient business processes, and maintaining the integrity of dependent applications [52]. Key dimensions of data quality include accuracy, completeness, consistency, timeliness, validity, and uniqueness [53]. However, achieving these standards is challenging due to factors like sensor accuracy and calibration issues, the sheer volume of data, integration complexities, transmission errors, latency, and security concerns [54].

The characteristics that are used to gauge the quality of the data are outlined in data quality dimensions [55]. Every company has a varying level of data quality, hence the metrics to be used vary as well. While several papers specify several of them, the following six are the most commonly used [53]:

- Accuracy: The information accurately reflects the "reality" that the organization is seeking to examine.
- Consistency: information is consistent, accurate, and logical throughout the organization's various datasets.

- Completeness: Every piece of information the business requires is readily accessible.
- Timeliness: The information must be available to the organization at the precise time that it is required.
- Uniqueness: A single, distinct piece of data only ever occurs in storage system.
- Validity: The data type and format match what is anticipated.

Once analyzed the six dimensions of data quality, a series of similarities can be observed among them. One of these similarities is found in dimensions of timeliness and uniqueness, which validate groups of values in datasets, they measure the quality of rows instead of unique values. The other similarity could be found in accuracy, completeness, consistency, and validity dimensions, that they quantify the quality of individual records of the datasets.

With the division of these groups, it can be clearly seen how dimensions that work with each record separately can be measured at the edge, while qualities that require general dataset knowledge or involve more than one record will be handled at the fog or cloud layer agent.

Record-level data quality dimensions

These dimensions focus on evaluating individual records for their quality, ensuring that each piece of data meets the required standards. They are suitable for edge processing, where data can be assessed independently and locally. This subsection explains the algorithms executed by the edge agent for quality monitoring purposes.

Algorithm 2 iterates through all defined consistency rules to verify whether the attributes of the record satisfy the specified requirements. These requirements are set during the agent's initialization and may include checks such as whether the values are logical for the measurement or whether they adhere to specified accuracy standards (e.g., the number of decimal places).

Algorithm 2 Consistency Dimension

Require: One record of m attributes, Consistency Rules R
Ensure: Consistency Score S

$inconsistencies \leftarrow 0$
$total\_checks \leftarrow 0$
$r_i \leftarrow record$
for each rule $r_j \in R$ **do**
    $total\_checks \leftarrow total\_checks + 1$
    if not $r_j(r_i)$ **then**
        $inconsistencies \leftarrow inconsistencies + 1$
    end if
end for
$S \leftarrow 1 - \frac{inconsistencies}{total\_checks}$
return S

The provided algorithm 3 calculates the accuracy score for a continuous stream of data by comparing each record with a Reference Data Model, D_{ref}. Initially, counters are initialized to keep track of the total number of processed records and how many of these records exactly match entries in D_{ref}. For each new record in data stream, it checks if it matches any record in D_{ref}, incrementing the corresponding counter upon a match. Subsequently, accuracy is calculated as the ratio of correctly matched records to the total number of processed records.

Algorithm 3 Accuracy Dimension

Require: One record of m attributes, Ref. Data Model D_{ref}
Ensure: Accuracy Score S

$total\_correct \leftarrow 0$
$total\_records \leftarrow 0$
$r_i \leftarrow record$
$total\_records \leftarrow total\_records + 1$
if r_i matches any record in D_{ref} **then**
    $total\_correct \leftarrow total\_correct + 1$
end if
$S \leftarrow \frac{total\_correct}{total\_records}$
return S

Algorithm 4 calculates the Completeness Score for a record in a data stream with m attributes. Initially, two variables are set: total_missing to count the number of missing attributes in current record and expected_total which represents the total expected number of attributes (assuming all attributes are expected to be present). The algorithm then iterates through each attribute A_j of the record and checks if it is absent. Each time it encounters a missing attribute, it increments the total_missing counter. After completing the iteration, it calculates the Completeness Score S as 1 minus the proportion of missing attributes relative to the expected total of attributes. Finally, it returns S, which indicates how complete the record is in terms of the presence of its attributes compared to what is expected.

Algorithm 4 Completeness Dimension

Require: One record of m attributes
Ensure: Completeness Score S

$total\_missing \leftarrow 0$
$expected\_total \leftarrow m$ ▷ Assuming all attributes are expected
for each attribute A_j in data **do**
    if A_j is missing **then**
        $total\_missing \leftarrow total\_missing + 1$
    end if
end for
$S \leftarrow 1 - \frac{total\_missing}{expected\_total}$
return S

The Validity Dimension algorithm 5 calculates the Validity Score for a record in a data stream containing m attributes, evaluating each attribute against a set of validity rules R. Initially, two variables are initialized:

$invalid\_count$ to count the number of attributes that do not comply with any validity rule, and total_checks to count the total number of checks performed.

For each attribute a_j in record r_i, the algorithm increments total_checks and checks if the attribute satisfies any of the rules specified in *R*. If the attribute does not satisfy any rule, *invalid count* is incremented. After reviewing all attributes of the record, the Validity Score is calculated as the proportion of invalid attributes relative to the total number of checks performed.

Algorithm 5 Validity Dimension

Require: One record of m attributes, Validity Rules V
Ensure: Validity Score S
$invalid\_count \leftarrow 0$
$total\_checks \leftarrow 0$
$r_i \leftarrow record$
for each attribute a_j in r_i **do**
    $total\_checks \leftarrow total\_checks + 1$
    if not a_j satisfies any rule in R **then**
        $invalid\_count \leftarrow invalid\_count + 1$
    end if
end for
$S \leftarrow 1 - \frac{invalid\_count}{total\_checks}$
return S

Row-level data quality dimensions

These dimensions assess the overall quality of the entire dataset, often requiring a broader view and cross-record analysis. They are better suited for fog or cloud environments, where more computational resources are available to handle complex evaluations. This subsection explains the algorithms executed agent for quality monitoring purposes.

Algorithm 6 calculates the Timeliness Score for a dataset D containing n records, each with a timestamp attribute T. Initially, two variables, max_timestamp and min_timestamp, are set to represent the maximum and minimum timestamp values found in dataset, respectively.

Algorithm 6 Timeliness Dimension

Require: Dataset D with n records and Timestamp T
Ensure: Timeliness Score S
$max\_timestamp \leftarrow x$
$min\_timestamp \leftarrow y$
for each record $r_i \in D$ **do**
    $timestamp \leftarrow T(r_i)$ ▷ Extract timestamp from record r_i
    if $timestamp > max\_timestamp$ **then**
        $max\_timestamp \leftarrow timestamp$
    end if
    if $timestamp < min\_timestamp$ **then**
        $min\_timestamp \leftarrow timestamp$
    end if
end for
Calculate timeliness score:
$range\_seconds \leftarrow max\_timestamp - min\_timestamp$
$current\_timestamp \leftarrow$ current time
$elapsed\_seconds \leftarrow current\_timestamp - min\_timestamp$
$S \leftarrow 1 - \frac{elapsed\_seconds}{range\_seconds}$
return S

Next, the algorithm iterates through each record r_i in D to determine the maximum and minimum observed timestamp values. For each record, the timestamp $T(r_i)$ is extracted. If *timestamp* is greater than max_timestamp, max_timestamp is updated; and if it is less than min_timestamp, min_timestamp is updated. After obtaining the maximum and minimum timestamp values in dataset, the Timeliness Score S is calculated. This is done by computing the time range in seconds between $max\_timestamp$ and $min\_timestamp$, and determining how much time has passed since the minimum timestamp value up to the current timestamp.

Algorithm 7 computes a uniqueness score S for a dataset D based on a specified unique identifier attribute U associated with each record. Initially, an empty set $unique\_set$ is created to store unique values. The algorithm iterates through each record $r_i$in D, extracting the unique identifier using the function $U(r_i)$ and adding it to $unique\_set$. After processing all records, the algorithm calculates $unique\_count$, which represents the number of unique identifiers in dataset. The uniqueness score S is then determined by dividing $unique\_count$ by the total number of records n.

Algorithm 7 Uniqueness Dimension

Require: Dataset D with n records and Unique Identifier U
Ensure: Uniqueness Score S

```
unique_set ← {}                          ▷ Initialize an empty set for tracking unique values
for each record r_i ∈ D do
    identifier ← U(r_i)                  ▷ Extract unique identifier from record r_i
    Add identifier to unique_set
end for
unique_count ← |unique_set|              ▷ Count the number of unique identifiers
S ← unique_count / n                     ▷ Calculate uniqueness score
return S
```

Multilayer deployment

The agent explained throughout this document has an advantage over other monitoring software on the market, which is its ability to run on the three known computing layers today. It can be seen in Table 1 that many existing agents do not comply with this requirement.

EdgeGuard is a comprehensive monitoring solution designed to adapt to the specific characteristics of each layer in the IoT architecture: edge, fog, and cloud. At the edge, monitoring prioritizes low power and processing consumption, which are essential for devices with limited resources. Here, key quality dimensions such as consistency, validity, completeness, and accuracy are measured (Section "Record-level data quality dimensions"), ensuring that the data generated is reliable from its origin while optimizing resource usage. Additionally, all the necessary metrics for the proper functioning of the edge, coupled with the need for the agent to be lightweight, make EdgeGuard ideal for working in low-processing environments. In contrast, at the fog and cloud layers, where greater computational power and energy capacity are available, the focus shifts to data uniqueness, preventing duplication in distributed systems, and timeliness, ensuring that data is processed promptly for critical decisions and complex analyses (Section "Row-level data quality dimensions"). Moreover, the solutions that this agent can provide are more complex than those that would be carried out at the edge. An example of what is monitored in these layers includes the processing load and the CPU and memory usage on the nodes, to ensure that the resources are adequate for the required data volume and processing.

Experimental results

In this section, the main results of the process carried out to validate the EdgeGuard monitoring agent are presented. The following sections cover the tests performed: (RQ1) the configuration to implement the different experiments, (RQ2) the scalability analysis of the agent, (RQ3) and (RQ4) the monitoring and forecasts carried out and (RQ5) the fulfillment of data quality measurement. The following research questions are established to serve as validation of the contributions listed in Section "Contributions":

RQ1 Is it possible to use the agent in diverse infrastructure devices without experiencing significant performance problems?

RQ2 As the number of computing continuum infrastructure devices increases, does the agent scale appropriately?

RQ3 Is the agent able to continuously monitor various metrics of heterogeneous infrastructural devices in computing continuum?

RQ4 Does the agent offer quality prediction metrics of the infrastructural devices so that potential issues can be anticipated before they occur?

RQ5 Is the agent able to evaluate the quality of the data so that it can be safely consumed?

RQ1 - Is it possible to use the agent in diverse infrastructure devices without experiencing significant performance problems?

We have modeled a scenario where a corporation is prepared to implement different AI life cycle processes in an Edge Computing environment [56], such as data pre-processing, AI model retraining in edge or AI model deployment and inference, in order to conduct the evaluation of the EdgeGuard solution under practical working settings. We explore how EdgeGuard can be applied in these complex scenarios to potentially enhance various device processes. Also, it can be especially helpful for monitoring the different computational layers that businesses can use (edge, fog and cloud), as well as improving the maintenance of these environments. This capability addresses F5, ensuring seamless deployment across the entire cloud continuum.

This scenario, which is shown in Table 3, pertains to an edge computing environment of a company in Spain, which aims to deploy AI lifecycle processes on edge devices to utilize the unused computing power. However, they face the challenge of needing to monitor each of the processes across the three existing computing layers (edge, fog, and cloud). Initially, the environment was formed by four (i) Raspberry Pi 4, three (ii) Google Coral Dev Boards, 2 (iii) Google Coral Mini and (iv) a NVIDIA Jetson Nano as the Edge layer. In addition, the Fog layer is composed of a pair of (v) Dell Precision 3520 and (vi) Precision 7560, and the Cloud layer is structured by three (vii) OpenStack [57] Virtual Machines and three (vii) on-premises virtual machines in company servers. The diversity of hardware and virtualization platforms in this setup serves to validate F8, proving the agent's compatibility with heterogeneous infrastructures.

In this edge computing environment, the company aims to perform different steps of the AI lifecycle on edge, fog, and cloud devices. By doing so, they can utilize the processing capabilities of elements that were previously only used for data collection to perform various tasks. These tasks include processing the collected data, retraining models with real-time data, or using the models alongside the data to perform inference and draw conclusions from the data. The problem they face is that the company is unable to easily monitor this environment due to the variety of devices and the challenge of deploying monitoring processes across different computing layers. Additionally, since these are AI lifecycle processes, the company is concerned about them stopping unexpectedly due to an error or lack of resources.

Terraform [58] has been used for the provisioning of these machines, and Ansible [59] is the configuration management tool chosen for installing and configuring the dependencies. The monitoring agent is installed in each infrastructure device and the monitored metrics are aggregated in a cloud database after the infrastructure specified in Table 3 has been provisioned and configured. These metrics are used to evaluate the proper functioning of the monitoring agent in Section "Experimental results".

RQ2 - As the number of computing continuum infrastructure devices increases, does the agent scale appropriately?

This subsection provides an analysis of EdgeGuard in various scenarios to determine its applicability across different devices throughout the multiple layers of computing. This evaluation has been carried out on four of the devices described in Table 3: a Google Coral Dev Board, a Raspberry Pi,a Dell 7560, and an OpenStack VM. The goal is to show how EdgeGuard is capable of maintaining proper functionality and low memory consumption across all the presented scenarios.

In accordance with F6, Table 4 provides a comprehensive statistical analysis of the computational resource demands of EdgeGuard on a Raspberry Pi 4, specifically examining the impact of varying process concurrency on RAM and CPU utilization. The empirical results demonstrate that memory consumption is characterized by remarkable stability; despite a significant increase in the number of concurrent processes, the mean RAM usage

Table 3 Experimental scenario of an infrastructure that covers the edge, fog and cloud

Type	Number of Devices	Layer	Memory (GiB)	CPU Cores	Disk (GiB)	OS
(i) Raspberry Pi 4	6	Edge	8.00	4	MicroSD	Raspberry Pi OS
(ii) Google Coral Dev Board	1	Edge	2.00	4	10.0+MicroSD	Mendel Linux
(iii) Google Coral Mini	2	Edge	1.00	4	5.0+MicroSD	Mendel Linux
(iv) NVIDIA Jetson Nano	1	Edge	4.00	4	60.0	Ubuntu 18.04
(v) Dell Precision 3520	2	Fog	16.00	4	512.0	Ubuntu 20.04
(vi) Dell Precision 7560	4	Fog	32.00	8	1024.0	Ubuntu 22.04
(vii) OpenStack VM	3	Cloud	24.0	16	500.0	Ubuntu 22.04
(viii) On-premise VM	3	Cloud	8.00	2	40.0	Ubuntu 22.04

Table 4 CPU and memory consumption of Edgeguard with varying number of processes on a raspberry Pi 4

	Processes	Min.	Max.	Mean	Std. Dev.
RAM (MB)	5	151.35	159.37	155.21	1.04
	7	153.65	158.23	154.54	0.99
	10	159.35	163.91	160.02	0.85
	12	150.53	162.81	157.28	2.53
	15	155.12	159.86	156.06	0.83
CPU (milicores)	5	74.83	261	133.67	37.19
	7	47.42	221.5	93.28	25.77
	10	86	261.25	129.99	35.56
	12	84.25	279.42	133.79	37.43
	15	75.92	272.08	121.38	35.34

remains confined within a narrow margin. This negligible scaling suggests a highly optimized memory management architecture suitable for resource-constrained environments. In contrast, CPU utilization exhibits greater volatility, with mean values showing less linear correlation to the process count and significant standard deviations that reflect the transient nature of processing loads. These fluctuations in CPU demand likely originate from the varying complexity of individual process tasks and the overhead associated with the operating system's dynamic scheduling of concurrent execution threads, rather than a fixed growth in baseline resource requirements.

In addition, Fig. 4 graph analyzes the performance of the agent in a period of two hours on different devices as the number of containers increases. The continuous tracking of these metrics over the specified time frame confirms the stability of F1. It is observed that the execution time of the agent grows almost linear with the number of containers, regardless of the hardware used, whether it is a development board, a powerful server, or a virtual machine.

The results of the experiment demonstrate a clear positive correlation between the number of monitored containers and the monitoring agent's execution time. To evaluate the system overhead in a realistic cloud-edge continuum scenario and ensure the results are easily replicable, the validation was conducted on four devices presented. The findings explicitly show that the computational overhead introduced by the agent remains remarkably low, even as the workload increases. Specifically, the execution time grows in an almost linear fashion relative to the number of containers, yet the absolute resource consumption confirms that the agent is exceptionally lightweight. These results prove that the framework can operate efficiently on resource-constrained devices without compromising system performance, highlighting its suitability for large-scale edge computing environments.

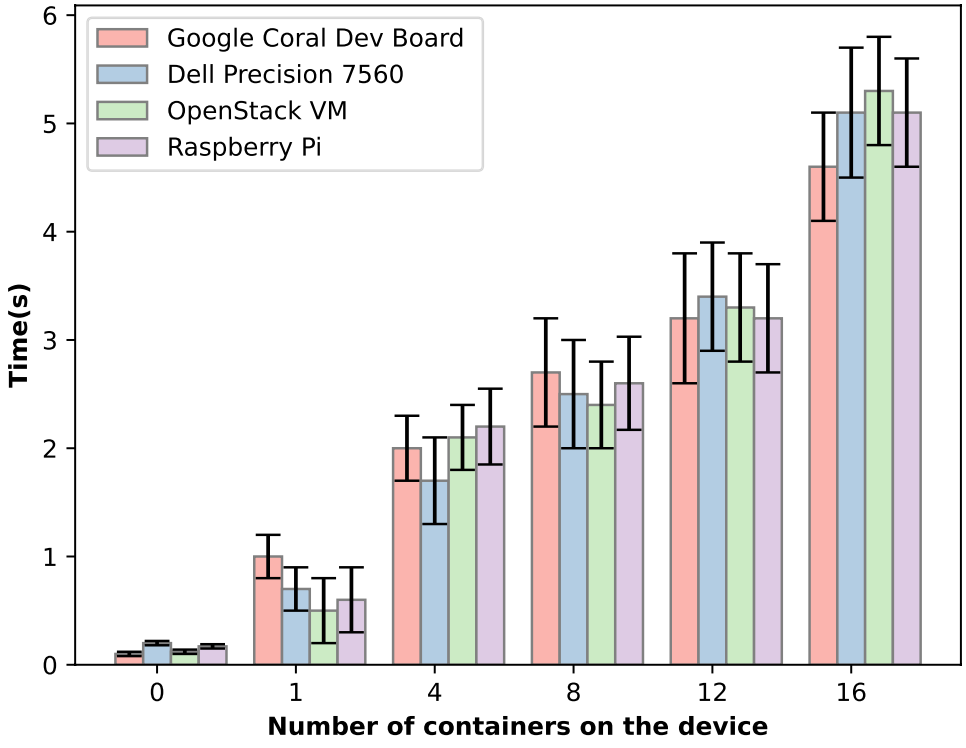

Fig. 4 Average execution times of EdgeGuard varying containers in different devices

RQ3 - Is the agent able to continuously monitor various metrics of heterogeneous infrastructural devices in computing continuum?

The aim of this subsection is to thoroughly evaluate the primary objectives of the EdgeGuard monitoring agent after it has been deployed continuously over a period of three months. This assessment seeks to analyze the agent's performance and efficacy across a range of monitoring tasks within specified environments, as outlined in Table 3. Each environment was chosen to simulate the conditions in which the EdgeGuard agent is expected to operate, encompassing edge, fog, and cloud layers to provide a comprehensive perspective on its adaptability and resilience.

To satisfy F4, the evaluation includes monitoring different system indicators, such as CPU usage, memory consumption, disk usage, and battery levels, to gain insights into the agent's consistency and responsiveness under varying workloads. By observing the agent's data collection accuracy, latency, and the efficiency of real-time updates across the three-tier architecture, this analysis aims to determine EdgeGuard's capacity to support high-stakes monitoring requirements and adapt to the heterogeneous nature of distributed systems.

For this experiment, data on CPU, RAM, battery, and disk space metrics were collected from a Dell Precision 3520. On this device, a task was run during the early morning hours that generated a high load on the device. in afternoon, this load was reduced, and during the night, the load was low. Additionally, the device performed write and delete operations on the disk to analyze this variable as well.

In addition to monitoring the status of the device, and as a core requirement of F2, the agent is equipped with a certain level of intelligence that can monitor the device's future state based on previously acquired knowledge. To achieve this prediction, the SNARIMAX model (Seasonal Non-linear AutoRegressive Integrated Moving-Average with eXogenous inputs) has been chosen.

Choosing a SNARIMAX model is beneficial in scenarios where time series exhibit complex, seasonal, and nonlinear patterns. SNARIMAX is ideal when there are exogenous variables that affect the behavior of the series, such as environmental factors or usage patterns, and when there is a desire to capture both long-term trends and seasonal fluctuations (like CPU or battery usage at different times of the day) [60]. This model also excels in predictive monitoring applications, as its ability to integrate both nonlinearities and seasonal factors allows for more accurate anticipation of significant changes and recurring patterns. Furthermore, it enables the modeling of behavioral patterns that vary according to context or time, which is critical for systems requiring constant and precise monitoring of resources and service quality [61].

Using the SNARIMAX model, predictions have been made that resulted in a series of representative graphs illustrating resource behavior over time. These graphs show complex patterns in CPU, RAM, disk and battery usage, reflecting both seasonal trends and nonlinear variations in demand. The forecast has been made for a horizon of two hours. in Fig. 5, the visualizations that illustrate these predictions are presented.

RQ4 - Does the agent offer quality prediction metrics of the infrastructural devices so that potential issues can be anticipated before they occur?

As a quantitative validation of F2, the Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) graphs are the three main prediction error metrics that are shown in this section. These metrics give a thorough evaluation of the agent's prediction accuracy and consistency in comparison to real data, revealing whether the agent is able to foresee any problems with infrastructure devices before they arise.

The average of the absolute disparities between the expected and actual values is known as the MAE. To emphasize general variations and give a baseline understanding of the agent's prediction accuracy, lower MAE values show that the agent's predictions are, on average, close to the real values [62, 63]. The RMSE reflects the magnitude of prediction errors, as it squares each difference between predicted and actual values, thus giving more weight to larger errors. A lower RMSE signifies that errors in predictions remain small, which is essential for early issue detection, as larger errors could signal emerging problems [63, 64]. Meanwhile, the MAPE provides a relative measure of prediction accuracy by calculating mistakes as a percentage of the actual values. This metric is particularly useful for assessing the agent's capacity to anticipate issues across various types of infrastructure devices, irrespective of data scale [65]. MAPE provides a scale-independent picture of the error, in contrast to MAE or RMSE, which are scale-dependent and impacted by the size of the data. Because of this, it is particularly useful for evaluating how well models function across datasets with various sizes or units. [63, 66].

Since departures from predicted patterns may point to possible problems that need to be addressed, analyzing these metrics offers important insights into how consistently the devices behave over time. It is feasible to determine whether the agent provides trustworthy quality prediction metrics for the devices by monitoring changes in MAE, RMSE, and MAPE over time. Long-term increases in these measures may indicate regions that require preventative maintenance, enabling any problems to be resolved before they become more serious.

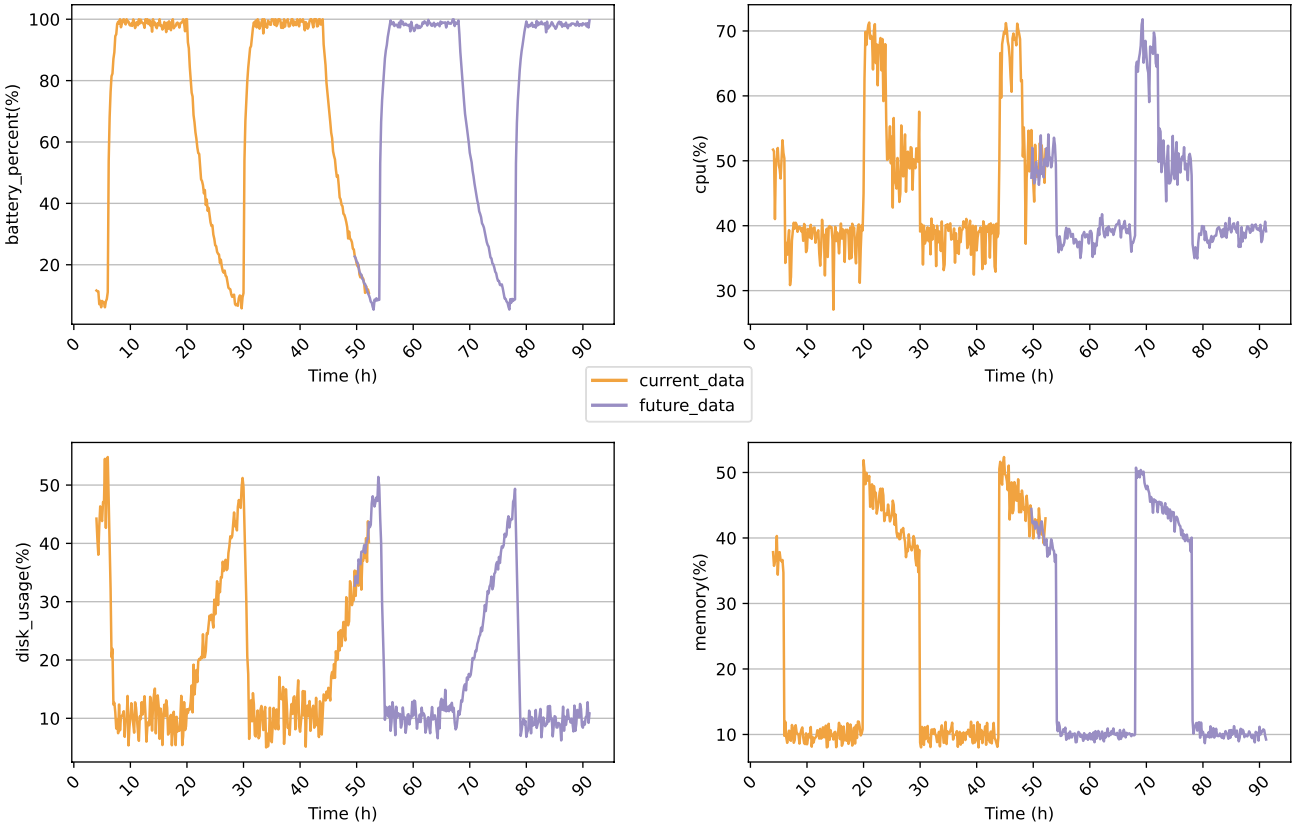

Fig. 5 Forecast of the monitored metric with SNARIMAX model

Each error metric is plotted over time in Fig. 6, which shows how prediction accuracy changes over the agent predictions and makes it possible to see any odd trends. The agent demonstrates strong forecasting accuracy and reliability for infrastructural devices, as evidenced by consistently low and stable MAE, RMSE, and MAPE values. The low, steady MAE indicates that the agent's predictions closely align with actual values, maintaining minimal consistent error. Similarly, the stable, low RMSE—without significant spikes—suggests that the agent avoids large errors in predictions, an essential quality for detecting potential issues early. Additionally, the low MAPE reflects precise predictions across varying device scales, confirming the agent's suitability for monitoring diverse infrastructure with proportional accuracy.

RQ5 - Is the agent able to evaluate the quality of the data so that it can be safely consumed?

This subsection answers research question 5 with the evaluation of the data quality experiment. This experiment is specifically designed to validate F3, consisting of measuring the quality indicators explained in Section "Data quality management" throughout the sending of two thousand jsons with ten values of temperature and

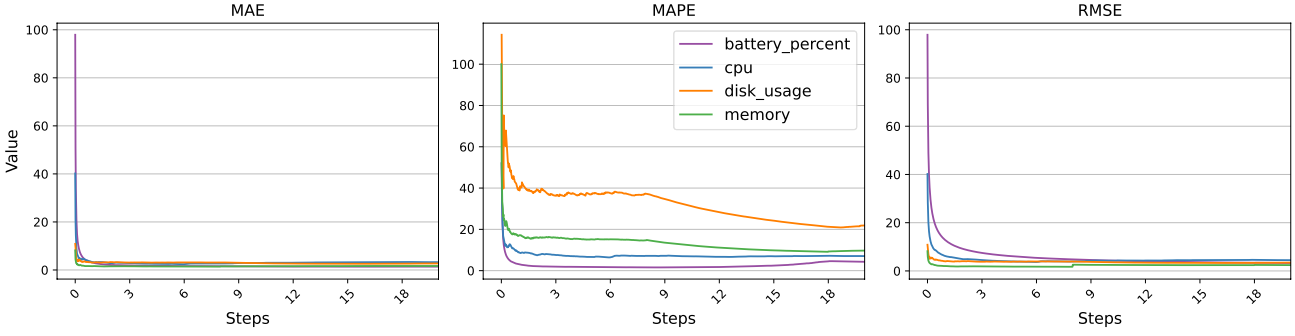

Fig. 6 Model accuracy assessment: MAE, MAPE and RMSE analysis for monitored metrics

Table 5 Statistical parameters of the data quality monitoring experiment

	Mean	Std.	25%	50%	75%	Kurtosis	Skew
Accuracy	84.76	8.72	79	85	91.25	−0.99	−0.05
Consistency	89.74	5.85	85	89.5	94.25	−1.09	0.058
Completeness	88.26	7.11	83	88	94	−1.03	−0.09
Timeliness	87.02	7.85	80	87.5	91	−1.28	0.06
Uniqueness	88.29	7,52	82	89	95	−1.19	−0.19
Validity	90.54	6.15	85.75	91	96	−1.07	−0.18

humidity metrics taken from a sensor. A series of errors have been added to this data to see if the agent is able to identify and measure them.

The Table 5 presents the statistical summaries of these evaluations, helping to determine if the data meets the quality thresholds to be deemed safe for consumption showing how the agent correctly measures the quality of the data.

The table demonstrates that the agent is capable of effectively measuring data quality by providing statistical indicators for each quality metric. These results indicate that the agent not only evaluates each dimension of quality but also provides detailed insights into the variability, trends, and distribution of each metric. Here's how each aspect of the table supports this conclusion:

- High means for each metric: The mean values for each indicator range between 84.76% and 90.54%, suggesting that the quality of the data used for the experiment had a low rate of quality errors.
- Low standard deviation: The standard deviations for each metric (between 5.85% and 8.72%) indicate stable measurements, with no major fluctuations. This shows that the experiment has no significant variations in its results.
- Percentiles and minimum/maximum limits: The percentile values and the minimum and maximum values show the distribution of quality measurements. The 50th percentile (median) and the 25th and 75th percentiles are close to the means, indicating that values are concentrated without much spread [67]. This implies that the experiment has a uniform level of quality, which matches with the input data.
- Kurtosis and skewness: The kurtosis and skewness of each metric are close to zero, suggesting that the distributions of the measurements are symmetrical and do not present extreme outliers [68]. This balance suggests that the experiment carried out has consistent results.

Expert evaluation

In this section, eight experts in operationalization of Cloud Continuum and Edge AI environments were pitted against EdgeGuard with the objective of monitoring a hybrid computing environment comprising edge, fog, and cloud. Two types of evaluations have been conducted to demonstrate the correct functioning and usefulness of EdgeGuard in enterprise market: an experiment in a fictitious scenario for quantitative evaluation and a series of questions to experts for qualitative evaluation.

Experiment setup

The experiment is framed within the following fictional scenario. "A company, which has a plant with different devices performing various periodic tasks, requires a trained individual to verify the correct functioning of its devices and, in case of failure, provide a quick solution. Therefore, the goal is to provide an analysis of each device as quickly as possible, without neglecting any, and offer a solution only when necessary." The experts have been asked to comply with the following statements:

- Access to dashboards in InfluxDB is granted to view the status of the devices over time.
- Access to the different infrastructure devices is granted to better understand the systems.

They have been asked to meet the following objectives:

- Identifying as many issues as possible that exceed the normal behavior of the device, which is explained below.
- Experts are also given the opportunity to make predictions about how they think the devices will behave in future.
- A detailed analysis of what is happening to the device will be provided, along with the proposed solution.

The environment consisted of the devices listed in Table 3. The battery, CPU, disk, and memory of each infrastructure device are monitored and gathered in a centralized database,and through dashboards, they were tasked with covering the highest number of alerts with the greatest accuracy and the lowest latency possible.

To make the experiment as close to reality as possible, a service has been deployed on each device to perform workload tests using the stress-ng library [69]. This service runs throughout the morning and simulates the execution of high CPU and memory consumption tasks,

Table 6 Comparison of the experts evaluation and EdgeGuard monitoring all experimental scenario devices

	A	P	R	S	J	Fβ	MCC	D
Expert 1	0,716	0,294	0,278	0,829	0,106	0,286	0,109	1:01:15
Expert 2	0,773	0,429	0,333	0,886	0,219	0,375	0,242	1:03:39
Expert 3	0,773	0,333	0,111	0,943	0,054	0,167	0,086	0:18:10
Expert 4	0,784	0,400	0,111	0,957	0,068	0,174	0,119	0:08:00
Expert 5	0,807	0,529	0,500	0,886	0,386	0,514	0,394	1:12:45
Expert 6	0,761	0,448	0,722	0,771	0,494	0,553	0,424	1:27:31
Expert 7	0,739	0,222	0,111	0,900	0,011	0,148	0,015	0:33:40
Expert 8	0,784	0,480	0,667	0,814	0,481	0,558	0,430	2:20:46
EdgeGuard	0,966	0,895	0,944	0,972	0,916	0,919	0,898	−1:55:00

as well as writing to the hard disk to reduce its capacity. The service executes these stress tests with variable time durations and load percentages, thus avoiding easily predictable behavior at first glance.

The study is structured as follows. A panel has been organized with the experts, where they were given a summary of the different devices under their responsibility and were instructed on the previously established guidelines to carry out a comparison with the EdgeGuard framework.

Measurement criteria

To evaluate the effectiveness of the experts against the automated system, the participants were required to continuously monitor the infrastructure and record specific operational observations. Their performance was measured based on the following activities:

- **Anomaly Detection:** Experts had to distinguish between transient performance spikes (caused by normal fluctuations) and genuine failures or bottlenecks induced by the stress-ng library.
- **Diagnostic Accuracy:** For every identified issue, experts were required to categorize the root cause (e.g., CPU exhaustion, memory leak, or disk saturation). Misidentifying the resource type was penalized in the final precision score.
- **Detection Latency:** Experts recorded the exact timestamp when they first noticed a deviation from normal behavior. This was compared against the actual start time of the stress service to calculate the response lag.
- **Predictive Foresight:** Beyond reactive monitoring, experts were tasked with identifying trends in the InfluxDB dashboards to predict if a device would reach a critical state within the next hour.
- **Solution Formulation:** For each detected alert, experts had to provide a brief technical resolution (e.g., process termination, resource reallocation, or hardware scaling), which was evaluated for technical feasibility.

Performance comparison: Edgeguard Vs. experts

Table 6 compares the results obtained by experts and the EdgeGuard system in evaluating devices in an experimental scenario. To measure the effectiveness of EdgeGuard and the experts, a series of evaluation metrics for machine learning present in these articles has been used [70–72]. A description of each metric and what the values mean is provided below:

- **Accuracy** (**A**): The percentage of both positive and negative predictions that are accurate out of all cases.
- **Precision** (**P**): A measure of the accuracy of positive predictions, expressed as the percentage of genuine positives among all anticipated positives.
- **Recall** (**R**): The proportion of true positives among all actual positives, measuring the ability to detect positive cases.
- **Specificity** (**S**): A measure of the capacity to identify negative cases, expressed as the percentage of true negatives among all actual negatives.
- **Youden's Index** (**J**): Recall + Specificity − 1, is a metric used to assess a test's capacity to differentiate across classes.
- **F-score** (**Fβ**): A measure of how well precision and recall are balanced, calculated as the harmonic mean of the two.
- **Matthews' Correlation Coefficient** (**MCC**): A valuable statistic for unbalanced classes, it assesses the quality of a binary classification considering all values of the confusion matrix. It has a scale of 0 (no better than random guessing) to + 1 (perfect prediction) and −1 (complete disagreement).
- **Mean Delay** (**D**): It is the typical amount of time it takes the system to identify an issue after it has arisen. A negative mean latency number suggests that the system anticipated the issue before it materialized.

In the results presented, EdgeGuard stands out as the system with the best performance across all evaluation metrics. Its accuracy (**A**), which measures the proportion of correct predictions, reaches 96.7%, significantly

outperforming the evaluated experts, whose accuracy ranges between 71.6% and 80.7%. It also demonstrates an excellent balance between precision (**P**, 89.5%) and recall (**R**, 94.4%), indicating its ability to correctly predict positive cases while consistently capturing true positives. The specificity (**S**) is also the highest at 97.2%, showing that EdgeGuard is superior in identifying negative cases compared to the experts.

With exceptional values of 0.916, 0.919, and 0.898, respectively, the Youden index (**J**), the F-score (**Fβ**), and the Matthews correlation coefficient (**MCC**) all support EdgeGuard's efficacy. The system's exceptional ability to discriminate between positive and negative cases is demonstrated by the Youden index, which exhibits an excellent balance between sensitivity and specificity. By finding a solid mix between recall and precision, EdgeGuard regularly performs well in identifying positive situations, according to the F-score. Meanwhile, the Matthews correlation coefficient, a robust and reliable solution, attests to the system's exceptional performance even in situations with unbalanced classes, with a value close to 0.9. To confirm the statistical significance of these results, a Wilcoxon Signed-Rank test was performed comparing the framework against the expert panel's average performance. The test yielded a p-value of 0.0156 ($W = 0$), indicating that EdgeGuard's superiority is statistically significant and highly unlikely to be the result of random variation.

Furthermore, EdgeGuard's average detection time (**D**) is still a crucial component; at −1:55:00, it can predict issues almost two hours before they arise. This is a significant advantage over the experts, who only notice issues after they have already materialized, as seen by their all-positive detection times. Because of its exceptional performance, EdgeGuard is a highly effective tool for proactive incident management.

Expert assessment: feasibility and effectiveness of EdgeGuard

In addition to the previous experiment, the following questions were asked to experts in field to corroborate the importance of using EdgeGuard in environments with a large number of devices. All questions were rated on a scale from 1: Strongly Disagree to 5: Strongly Agree:

Q1 Do you think it would be viable to use EdgeGuard on diverse infrastructure devices without encountering significant performance issues?

Q2 Do you believe that EdgeGuard is helpful for monitoring a larger number of metrics, such as network metrics or data quality metrics?

Q3 After the experiment, do you consider the use of EdgeGuard necessary not only for monitoring devices but also for performing predictive maintenance?

Q4 Do you find EdgeGuard to be helpful in maintaining the reliability of your organization's computing devices?

Q5 Do you find the implementation of EdgeGuard in your organization to be simple?

To ensure the content validity of the expert assessment, the Content Validity Ratio (CVR) was calculated for each item based on Lawshe's method [73]. The panel consisted of eight experts ($N = 8$), a size that aligns with the recommendations of Lynn [74], who suggests that a group between five and ten members is optimal for providing sufficient content validation without diminishing returns. Given that the experts rated the items on a 5-point Likert scale, ratings of 4 ("Agree") and 5 ("Strongly Agree") were categorized as "Essential" to determine the n_e value. With $N = 8$, the minimum critical CVR value required to ensure that the agreement is not due to chance is 0.75 at a confidence level of $\alpha = 0.05$. The analysis showed that all evaluated items consistently exceeded this threshold, with CVR values ≥ 0.75, and a global Content Validity Index (CVI) of 0.90. Furthermore, the Interquartile Range (IQR) for all questions remained between 0 and 1, confirming a high degree of consensus among the experts and low dispersion in their ratings. This ensures that the results reflect a genuine representation of the construct's validity rather than random agreement or polarized opinions.

Once all the experts answered these questions, Fig. 7 response chart was obtained. The positive feedback indicate that EdgeGuard achieves its core goals and emphasizes its value in a range of computing continuum infrastructure applications. Experts pointed out that it's lightweight architecture makes it possible to deploy it effectively on devices with minimal processing power, guaranteeing peak performance without causing major problems. In heterogeneous edge environments, where a variety of hardware necessitates extremely flexible solutions, this characteristic is essential. They also highlighted its modular design and ease of integration, which enable deployment across various environments such as virtual machines, containers, and bare-metal configurations. It is the perfect tool for handling complicated and large-scale deployments because of this and its capacity for horizontal scaling. Its long-term sustainability is guaranteed by this scalable strategy, which also establishes it as a reliable option for edge ecosystems that are expanding steadily.

Furthermore, experts highlighted real-time continuous monitoring as one of its most relevant strengths, as

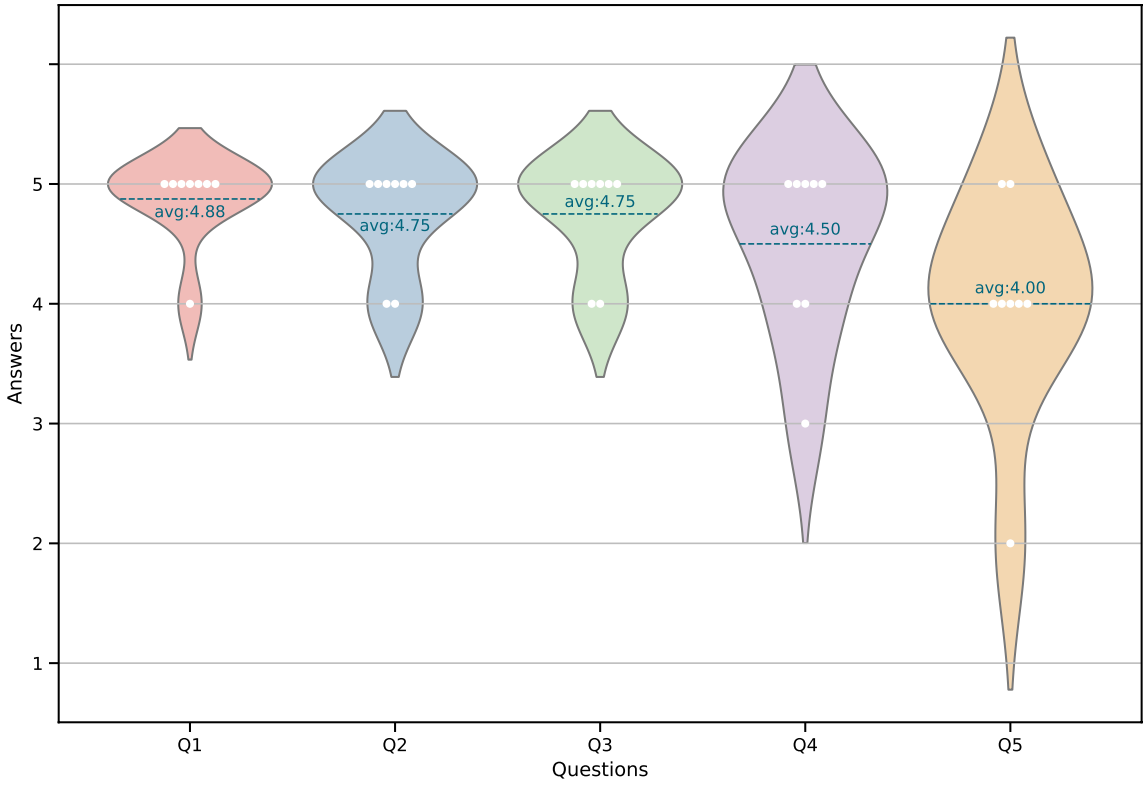

Fig. 7 Expert assessment: insights and responses from the experiment

it provides detailed information on the status of devices, network metrics, and application performance. They also valued the integration of advanced machine learning algorithms, which enable the detection of anomalies and predictions of potential failures, thereby facilitating predictive maintenance and reducing downtime. Proactive reactions and resilience to new problems are ensured by this capacity for ongoing learning and pattern adaption. These insights solidify EdgeGuard as an appropriate tool not only for monitoring and predictive maintenance but also for ensuring data quality, making it a comprehensive and highly valuable solution for modern edge environments.

Discussion

In this research, we address the challenges of monitoring distributed, heterogeneous environments across edge-cloud continuum layers. To do this, we introduce EdgeGuard, a lightweight and adaptable framework made to provide dependable, real-time monitoring while working within limitations of edge computing. The framework seeks to close the gap between the resource constraints that are specific to edge environments and the various requirements of edge applications.

The framework's design focuses on achieving several key objectives. Firstly, it provides a robust mechanism for processing, managing, and interpreting diverse data streams, enabling efficient monitoring of distributed systems. This process begins with defining a well-structured and adaptable data schema, which acts as the foundation for all monitoring operations, as detailed in Section "Agent data schemas". The schema ensures interoperability across different devices and platforms while maintaining the flexibility to adapt to evolving requirements. Furthermore, the advanced algorithms perform state prediction by analyzing patterns and trends within collected data, allowing for proactive decision-making and incident prevention, discussed in Section "Prediction and status check".

In addition to monitoring, the framework also addresses the quality of the data being processed directly on edge devices, an often-overlooked aspect of edge computing systems. By evaluating and ensuring the quality of data at its source, EdgeGuard minimizes the risk of propagating errors or inconsistencies across the network layers. This capability is particularly important in environments where decisions are made based on real-time data streams, and it is described in detail in Section "Data quality management".

Another critical feature is its ability to seamlessly operate across the different existing layers without interfering with the functionality of applications deployed on these infrastructures. The implementation details and

strategies for achieving this are outlined in Section "Multilayer deployment".

Next, using the standards listed in Table 1, we provide an assessment of EdgeGuard's suitability as a tool for heterogeneous distributed environment monitoring.

F1 Real-Time Monitoring: In Section "Prediction and status check", live data collecting and visualization experiments show off EdgeGuard's real-time system metrics and performance monitoring capabilities, showcasing instant updates on several levels. The efficiency of real-time monitoring in identifying and resolving problems quickly is further supported by expert comments.

F2 Forecast Monitoring: This innovative capability, discussed in Section "Data quality management", highlights the system's ability to forecast future patterns and optimize resources using predictive algorithms. Trials confirm its effectiveness in making proactive decisions, and expert assessments recognize its potential to enhance the reliability of organizational computing devices.

F3 Data Quality: Sectiion "Data quality management" demonstrates the fulfillment of data quality requirements through metrics. Experiments conducted with EdgeGuard validate its ability to maintain high-quality data flows (Section "Experimental results"), supported by expert reviews affirming the robustness of the approach

F4 Multi-Level monitoring: EdgeGuard provides comprehensive insights across five distinct monitoring levels in Section "Agent data schemas". At the machine level, it tracks critical metrics of devices. The container level focuses on resource utilization of docker containers. The end-to-end link level measures network performance. At the application level, it focuses on monitoring the performance and health of applications running within the environment. Finally, the data level ensures data quality, along with monitoring data flow volume. When combined, these levels make it possible to provide a comprehensive strategy for performance optimization and monitoring in cloud continuum environments.

F5 Multi-layer: The ability to seamlessly deploy across various devices in edge, fog, and cloud environments ensures flexibility and adaptability in heterogeneous systems. As explained in Section "Multilayer deployment" and demonstrated through experiments and expert feedback (Section "Expert evaluation"), multi-layer deployment ensures comprehensive coverage across diverse infrastructures, enabling consistent performance and monitoring regardless of the deployment context. This highlights its scalability and effectiveness in real-world scenarios.

F6 Scalability: As described in Section "Experimental results", the agent can obtain required metrics from the heterogeneous distributed infrastructure. Furthermore, a thorough analysis of EdgeGuard's scalability needs in terms of memory and time consumption is provided in Section "Experimental results".

F7 Open-source: The agent system is openly accessible on GitHub, and EdgeGuard has been made available under the Apache License [75].

F8 Technology agnostic: The numerous issues that may occur with the various devices are not attempted to be resolved by EdgeGuard. As a result, it can work alongside other tools that are better suited for each of the several functions that make up the cloud continuum ecosystem.

In Section "Contributions", the contributions of EdgeGuard and the entire study were previously discussed. We highlight in the following list how the agent has fulfilled these contributions in addition to highlighting them throughout the text.

C1 The agent provides significant customization options for monitoring, as described in Section "Agent data schemas", enabling precise configuration for each device based on specific needs. The importance of this scalable monitoring capability was further emphasized in question Q2, where all experts agreed that the agent's adaptability is a key factor in its effectiveness.

C2 The agent's capability to measure data quality is presented in Section "Data quality management" and demonstrated in Section "Experimental results", where it evaluates key indicators such as accuracy, completeness, and consistency across data captured by different devices. The results confirm the agent's ability to assess data quality effectively at the edge. By leveraging these indicators, the system generates alerts in real-time, enabling proactive monitoring of data integrity. This functionality ensures that potential issues are identified and addressed promptly, maintaining high-quality data streams.

C3 Through the experiment described in Section "Experimental results", it has been demonstrated that the agent runs efficiently on a wide variety of devices with different specifications without impacting their computational power. Furthermore, the scalability of the agent, as addressed in Section "Experimental results", shows that its resource consumption remains low, and its response time is not high, further supporting its viability in diverse environments. This was confirmed by the experts in question Q1 posed in Section "Expert assessment: feasibility and effectiveness of EdgeGuard", where all agreed that its

use in such environments is viable due to its low resource consumption.

C4 By analyzing incoming data, Section "Experimental results" presents the predictions generated by the agent using machine learning algorithms, showcasing its capability to monitor and to forecast the future status of devices accurately. In Section "Experimental results", the validation process confirms the correctness of these predictions by comparing them with actual observed data, highlighting the system's reliability. Through real-time anomaly detection, the agent effectively identifies deviations from normal behavior, as seen in reported cases. Additionally, the experiment demonstrated that the agent outperformed all the experts in measuring device errors (Section "Performance comparison: edgeGuard vs. experts"). This led the experts to conclude that EdgeGuard is useful not only for monitoring but also for performing predictive maintenance on devices (Q3 in Section "Expert assessment: feasibility and effectiveness of EdgeGuard").

C5 This contribution is supported by the reference architecture described in Section "Architecture overview", which operationalizes the proposed concepts into a structured and coherent design. The architecture illustrates how the monitoring plane collects and correlates metrics across multiple layers while the repair plane integrates adaptive mechanisms to trigger corrective or preventive actions.

To sum up, EdgeGuard is a useful tool for resource monitoring and management in cloud continuum environments, especially when low latency, high availability, and flexibility are critical. It shines at offering thorough visibility of dispersed resources while maintaining steady performance even in dynamic environments since it is built to function flawlessly in diverse ecosystems. Its strategy not only ensures adherence to the unique limitations and specifications of every environment, but it also maximizes resource use, fostering sustainability and operational effectiveness. Additionally, by integrating it with predictive capabilities and advanced analytics systems, enterprises may improve decision-making, foresee problems, and guarantee that service-level agreements (SLAs) are met [76]. Furthermore, EdgeGuard has outperformed seasoned experts in field of edge resource management, showcasing its ability to deliver superior results across various operational challenges in Table 6.

Conclusions and future work

In this research we propose EdgeGuard, an advanced intelligent agent designed to facilitate operations teams in deployment and management of distributed applications within heterogeneous ecosystems. It offers real-time monitoring and predictive capabilities to assess the future state of devices and systems, enabling proactive decision-making and reducing downtime. It measures the quality of data processed by devices, ensuring reliable and actionable insights that enhance the overall effectiveness of data-driven operations. Furthermore, it is designed for seamless deployment across multi-layer environments, including edge, fog, and cloud, providing scalability with low resource consumption and rapid execution. This efficiency is maintained even as the number of monitored metrics increases, making it a robust solution for managing complex and dynamic workloads in modern distributed systems.

To assess the system's efficacy, we devised a comprehensive scenario to evaluate EdgeGuard's scalability, predictive accuracy, and data quality measurement capabilities. Experimental results confirm that it is capable of scaling effectively with respect to both time and memory across varying workloads, maintaining performance and reliability. Additionally, its ability to predict future states of devices and systems was rigorously tested, demonstrating a high degree of accuracy. Furthermore, the system's capacity to measure the quality of data processed by devices was evaluated, ensuring that data-driven operations are underpinned by reliable and consistent datasets. To further evaluate its practical utility, EdgeGuard was benchmarked against expert-driven orchestration methods through a targeted error monitoring experiment. These tests revealed that not only matches but often surpasses expert performance in monitoring precision, underscoring its suitability for real-world applications. Due to this, we have determined that it is suitable for the case under investigation since, in contrast to the other options under consideration, it can satisfy the requirements listed in Table 1.

Future research will focus on enhanced monitoring of device integrity and data security to ensure robust operation in sensitive environments. Advancements in predictive algorithms will improve resource allocation and enable dynamic adaptation to evolving workloads. The system will become part of a larger distributed management framework, incorporating tools for on-device diagnostics and issue resolution to reduce external dependencies. Additionally, new metrics will be integrated to optimize energy efficiency, supporting sustainable and environmentally conscious computing practices, promoting adherence to green computing principles. A key direction will also be the implementation and validation of the proposed reference architecture in real cloud continuum environments, enabling practical assessment of its effectiveness and adaptability under heterogeneous and dynamic conditions.

Author contribution

Lander Bonilla Viana: Conceptualization, Software, Formal analysis, Validation, Investigation, Visualization, Writing - Original Draft, Writing - Review & Editing.

Josu Diaz-de-Arcaya: Conceptualization, Funding acquisition, Methodology, Investigation, Writing - Review & Editing, Validation, Supervision. Juan López-de-Armentia: Writing - Review & Editing, Validation. Aitor Almeida: Supervision, Writing - Review & Editing, Funding acquisition.

Funding
This work was supported by NexusForum.EU project, that is a Coordination & Support Action co-funded by the European Union's Horizon Europe research and innovation programme under Grant Agreement 101,135,632 and by the Swiss State Secretariat for Education, Research, and Innovation (SERI). The objective of NexusForum.EU is to boost the consolidation of the European Computing Continuum ecosystem, as well as to provide a forward-looking and bold vision in new areas and directions that have not been explored so far.

Data availability
Data will be made available upon request.

Material availability
Material will be made available upon request.

Code availability
Code will be made available upon request.

Declarations

Ethics approval and consent to participate
It has been verified that the manuscript complies with the integrity and good scientific practice guidelines set by the journal.

Consent for publication
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 19 September 2025 / Accepted: 27 February 2026
Published online: 19 March 2026

References

1. Alberti E, Alvarez-Napagao S, Anaya V, Barroso M, Barrué C, Beecks C, Bergamasco L, Chala SA, Gimenez-Abalos V, Graß A et al (2024) Ai lifecycle zero-touch orchestration within the edge-to-cloud continuum for industry 5.0. Systems 12(2):48
2. McAfee A, Brynjolfsson E, Davenport TH, Patil D, Barton D (2012) Big data: the management revolution. Harv Bus Rev 90(10):60–68
3. Medeiros MMD, Hoppen N, Maçada ACG (2020) Data science for business: benefits, challenges and opportunities. The Bottom Line 33(2):149–163
4. Rojas E, Carrascal D, Lopez-Pajares D, Manso N, Arco JM (2024) Towards AI-enabled cloud continuum for iiot: challenges and opportunities. In 2024 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA), IEEE, pp 1–6
5. Boda VVR (2024) Edge computing in healthcare: what it is and why it matters. MZ Comput J 5(2)
6. Kuchuk H, Malokhvii E (2024) Integration of iot with cloud, fog, and edge computing: a review. Adv Inf Syst 8(2):65–78
7. Singh R, Gill SS (2023) Edge AI: a survey. Internet Things Cyber-Phys Syst 3:71–92
8. Gong T, Zhu L, Yu FR, Tang T (2023) Edge intelligence in intelligent transportation systems: a survey. IEEE Trans Intell Transp Syst Intell Transp Syst 24(9):891–894
9. Singh R, Chawla P, Gill SS (2025) Computational intelligence-based carbon neutral wireless networks for edge-cloud continuum. In: Computational intelligence techniques for 5G enabled IoT networks. Springer, pp 3–15
10. Yang J, Baker T, Gill SS, Yang X, Han W, Li Y (2022). A federated learning attack method based on edge collaboration via cloud. Software: practice and experience
11. Brandón A, Pérez MS, Montes J, Sanchez A (2018) Fmone: a flexible monitoring solution at the edge. Wireless Commun Mob Comput 2018:1–15
12. Taherizadeh S, Jones AC, Taylor I, Zhao Z, Stankovski V (2018) Monitoring self-adaptive applications within edge computing frameworks: a state-of-the-art review. J Syst Softw 136:19–38
13. Alwakeel AM (2021) An overview of fog computing and edge computing security and privacy issues. Sensors 21(24):8226
14. Yahuza M, Idris MYIB, Wahab AWBA, Ho AT, Khan S, Musa SNB, Taha AZB (2020) Systematic review on security and privacy requirements in edge computing: state of the art and future research opportunities. IEEE Access. 8:76541–76567
15. Mohanta BK, Jena D, Satapathy U, Patnaik S (2020) Survey on iot security: challenges and solution using machine learning, artificial intelligence and blockchain technology. Internet Things 11:100227
16. Alwarafy A, Al-Thelaya KA, Abdallah M, Schneider J, Hamdi M (2020) A survey on security and privacy issues in edge-computing-assisted internet of things. IEEE Internet Things J 8(6):4004–4022
17. Pezzullo GJ, Di Martino B (2024) Artificial intelligence techniques for dynamic offloading in cloud continuum environment: a review. In International Conference on Complex, Intelligent, and Software Intensive Systems, Springer, pp 405–412
18. He Z, Li K, Li K, Zhou W (2021) Server configuration optimization in mobile edge computing: a cost-performance tradeoff perspective. Softw Pract Exp 51(9):1868–1895
19. Diaz-De-Arcaya J, Torre-Bastida AI, Zárate G, Miñón R, Almeida A (2023) A joint study of the challenges, opportunities, and roadmap of mlops and aiops: a systematic survey. ACM Comput Surv 56(4):1–30
20. Alghamdi I, Anagnostopoulos C, Pezaros DP (2021) Data quality-aware task offloading in mobile edge computing: an optimal stopping theory approach. Futu Gene Com Sys 117:462–479
21. Bonilla L, López Osa MJ, Diaz-de-Arcaya J, Torre-Bastida AI, Almeida A (2024) Purity: a new dimension for measuring data centralization quality. In Proceedings of the 2024 8th International Conference on Cloud and Big Data Computing, pp 8–14
22. Akbari N, Toosi AN, Grundy J, Khalajzadeh H, Aslanpour MS, Ilager S (2024) Icontinuum: an emulation toolkit for intent-based computing across the edge-to-cloud continuum. In 2024 IEEE 17th International Conference on Cloud Computing (CLOUD), pp 468–474). IEEE
23. Babar M, Sohail Khan M (2021) Scaledge: a framework for scalable edge computing in internet of things–based smart systems. Int J Distrib Sens Netw 17(7):15501477211035332
24. Patel YS, Townend P, Singh A, Östberg P-O (2024) Modeling the green cloud continuum: integrating energy considerations into cloud–edge models. Clust Comput 27(4):4095–4125
25. Kazi KSL (2024) Artificial intelligence (AI)-driven iot (aiiot)-based agriculture automation. In: Advanced computational methods for agri-business sustainability. IGI Global, pp 72–94
26. Ltd., T (2024) MONIT barking at daemons. https://mmonit.com/monit/. Accessed 17 Apr 2024
27. Großmann M, Klug C (2017) Monitoring container services at the network edge. 2017 29th Int Teletraffic Congr (ITC 29) 1:130–133. IEEE
28. Korontanis I, Makris A, Theodoropoulos T, Tserpes K (2023) Real-time monitoring and analysis of edge and cloud resources. In Proceedings of the 3rd Workshop on Flexible Resource and Application Management on the Edge, pp 13–18
29. Inc., Z (2024) Zenoss, full-stack monitoring. Observability AIOps. https://www.zenoss.com/. Accessed 18 Apr 2024
30. Team GD (2024) What is ganglia? http://ganglia.info. Accessed 18 Apr 2024
31. Z LLC (2024) The all-in-one, open-source solution that lets you monitor anything. https://www.zabbix.com/. Accessed 18 Apr 2024
32. Nagios Enterprises L (2024) Nagios the industry standard in IT infrastructure monitoring. https://www.nagios.org/. Accessed 18 Apr 2024
33. Systems O (2023) The open source cloud & edge computing platform. Accessed 2024–04–18. https://opennebula.io/
34. De Chaves SA, Uriarte RB, Westphall CB (2011) Toward an architecture for monitoring private clouds. IEEE Commun Mag 49(12):130–137
35. Povedano-Molina J, Lopez-Vega JM, Lopez-Soler JM, Corradi A, Foschini L (2013) Dargos: a highly adaptable and scalable monitoring architecture for multi-tenant clouds. Futu Gene Com Sys 29(8):2041–2056
36. Clayman S, Galis A, Chapman C, Toffetti G, Rodero-Merino L, Vaquero L, Nagin K, Rochwerger B (2010) Monitoring service clouds in the future internet. pp 115–126. https://doi.org/10.3233/978-1-60750-539-6-115

37. Trihinas D, Pallis G, Dikaiakos MD (2014) Jcatascopia: monitoring elastically adaptive applications in the cloud. In 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp 226–235. IEEE
38. Miglierina M, Di Nitto E (2016) Monitoring in a multi-cloud environment. In: Model-driven development and operation of multi-cloud applications: the MODAClouds approach. Springer, Cham, Switzerland, pp 47–52. https://doi.org/10.1007/978-3-319-46031-4
39. Ab QI (2024) Talend | a complete, scalable data management solution. Accessed:2024–04–25. https://www.talend.com/
40. Google: OpenRefine (2024). https://openrefine.org/. Accessed 25 Apr 2024
41. IBM (2024) IBM® InfoSphere information server. https://www.ibm.com/es-es/information-server. Accessed 25 Apr 2024
42. Shi W, Cao J, Zhang Q, Li Y, Xu L (2016) Edge computing: vision and challenges. IEEE Internet Things J 3(5):637–646
43. Aruna K, Pradeep G (2020) Performance and scalability improvement using iot-based edge computing container technologies. SN Comput Sci 1(2):91
44. Engelfriet A (2009) Choosing an open source license. IEEE Softw 27(1):48–49
45. Beattie Z, Miller LM, Almirola C, Au-Yeung W-TM, Bernard H, Cosgrove KE, Dodge HH, Gamboa CJ, Golonka O, Gothard S et al (2020) The collaborative aging research using technology initiative: an open, sharable, technology-agnostic platform for the research community. Digit Biomarker 4(Suppl 1):100–118
46. Rochwerger B, Breitgand D, Levy E, Galis A, Nagin K, Llorente IM, Montero R, Wolfsthal Y, Elmroth E, Caceres J et al (2009) The reservoir model and architecture for open federated cloud computing. IBM J Res Devel 53(4):4–1
47. Inc. I (2024) InfluxDB. It's about time. Accessed 2024-04-25. https://www.influxdata.com/
48. Davis C (2021) Graphite. Accessed 2024-04-25. https://graphiteapp.org/
49. Group DUW (2013) The six primary dimensions for data quality assessment, defining data quality dimensions
50. Maxhalford (2024) River, online machine learnig in Python. Accessed 2024–11–12. https://riverml.xyz/latest/
51. Montiel J, Halford M, Mastelini SM, Bolmier G, Sourty R, Vaysse R, Zouitine A, Gomes HM, Read J, Abdessalem T et al (2021) River: machine learning for streaming data in python. J Mach Learn Res 22(110):1–8
52. Klein A, Do H-H, Hackenbroich G, Karnstedt M, Lehner W (2007) Representing data quality for streaming and static data. In 2007 IEEE 23rd International Conference on Data Engineering Workshop, pp 3–10. IEEE
53. Bhandari S, Ranjan N, Kim Y-C, Park J-D, Hwang K-I, Kim W-H, Hong Y-S, Kim H (2021) An automatic data completeness check framework for open government data. Appl Sci 11(19):9270
54. Karkouch A, Mousannif H, Al Moatassime H, Noel T (2016) Data quality in internet of things: a state-of-the-art survey. J Educ Chang Network Comput Appl 73:57–81
55. ISO/IEC (2022) ISO/IEC 25012. https://iso25000.com/index.php/en/iso-25000-standards/iso-25012. Accessed 23 Jan 2024
56. De Silva D, Alahakoon D (2022) An artificial intelligence life cycle: from conception to production. Patterns 3(6)
57. Foundation O (2024) OpenStack.OpenStack. The most widely deployed open source cloud software in the World. https://www.openstack.org/. Accessed 12 Nov 2024
58. HashiCorp T (2024) Terraform. https://www.terraform.io/. Accessed 12 Nov 2024
59. Hat R (2024) Ansible collaborative. https://www.ansible.com/. Accessed 12 Nov 2024
60. Wette S, Heinrichs F (2024). Oml-ad: online machine learning for anomaly detection in time series data. arXiv preprint arXiv:2409.09742
61. Nesse MB (2024) Forecasting inflation in Norway using machine learning. Master's thesis, Norwegian University of Life Sciences
62. Li Z, Rao Z, Pan L, Wang P, Xu Z (2023) Ti-mae: self-supervised masked time series autoencoders. arXiv preprint arXiv:2301.08871
63. Chicco D, Warrens MJ, Jurman G (2021) The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation. PeerJ Comput Sci 7:623
64. Du Y, Wang J, Feng W, Pan S, Qin T, Xu R, Wang C (2021) Adarnn: adaptive learning and forecasting of time series. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp 402–411
65. Aslanpour MS, Gill SS, Toosi AN (2020) Performance evaluation metrics for cloud, fog and edge computing: a review, taxonomy, benchmarks and standards for future research. Internet Things 12:100273
66. Azan ANAM, Mototo NFAMZ, Mah PJW (2021) The comparison between arima and arfima model to forecast kijang emas (gold) prices in Malaysia using mae, rmse and mape. J Comput Res Innov 6(3):22–33
67. Birpınar ME, Kızılöz B, Şişman E (2023) Classic trend analysis methods paradoxical results and innovative trend analysis methodology with percentile ranges. Appl Climatol 153(1):1–18
68. Busse A, Jelly T (2023) Effect of high skewness and kurtosis on turbulent channel flow over irregular rough walls. J Turbul 24(1–2):57–81
69. ColinIanKing (2024) Stress-ng (stress next generation). https://github.com/ColinIanKing/stress-ng. Accessed 09 Dec 2024
70. Chen J-H, Hsu S-C, Shen C-Y, Shieh G-S, Yang C-H, Kuo Y-M (2024) Deep-learning based classification of clinical significance for prostate cancer. In 2024 International Conference on Consumer Electronics-Taiwan (ICCE-Taiwan), IEEE, pp 321–322
71. Alzyoud M, Alazaidah R, Aljaidi M, Samara G, Qasem M, Khalid M, Al-Shanableh N (2024) Diagnosing diabetes mellitus using machine learning techniques. Int J Multiling Data Network Sci 8(1):179–188
72. Rainio O, Teuho J, Klén R (2024) Evaluation metrics and statistical tests for machine learning. Sci Rep 14(1):6086
73. Lawshe CH (1975) A quantitative approach to content validity. Pers Psychol 28(4)
74. Lynn MR (1986) Determination and quantification of content validity. Nurs Res 35(6):382–386
75. Bonilla L (2025) Edgeguard. https://github.com/LanderBV/edgeguard. Accessed 27 Jan 2025
76. Bocianiak K, Pawlikowski T, Podlasek A, Wary J-P, Wierzbowski J (2024) Challenges for continuous, provable security service level agreement management in computing continuum. IEEE Access

Publisher's Note