Internet of Things 35 (2026) 101830

Contents lists available at ScienceDirect

Internet of Things

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

Research article

Collective intelligence-based service migration enabling zoom-in functionality within industry 5.0

Riccardo Venanzi [a,*], Lorenzo Colombi [b], Davide Tazzioli [a], Simon Dahdal [b], Mauro Tortonesi [b], Luca Foschini [a]

[a] DISI - Dipartimento di Informatica Scienze e Ingegneria, University of Bologna, Viale Risorgimento, 2, Bologna, Italy
[b] Big data & Compute Continuum Lab, University of Ferrara, Via Saragat, 1, Ferrara, Italy

ARTICLE INFO

Keywords:
Collective intelligence
Artificial intelligence
Industry 5.0
Service migration
Zoom-In functionality
Industrial internet of things
Kubernetes

ABSTRACT

The rapid evolution of Industry 5.0 emphasizes the integration of human expertise with machine intelligence to create resilient, adaptive, and human-centric industrial systems. This paper introduces a novel Collective Intelligence (CI)-based service migration framework designed for Industry 5.0 environments, enabling dynamic orchestration of stateful services across heterogeneous edge-to-cloud infrastructures. At its core, the framework leverages Kubernetes (K8s) enhanced with AI-driven decision-making and human-in-the-loop collaboration to address the limitations of traditional orchestration in industrial settings. A key innovation of this work is the Zoom-In functionality, which empowers human operators to escalate anomaly detection and analysis by deploying advanced machine learning models on demand, seamlessly migrating services to resource-rich nodes when deeper investigation is warranted. The proposed framework integrates Large Language Models (LLMs) to translate operator intent into actionable policies, ensuring context-aware and explainable decision-making. Experimental validation in real industrial scenarios demonstrates high anomaly detection accuracy (F1-scores up to 1.0), reliable operator intent translation (over 70 % correct JSON generations with lightweight LLMs), and efficient multi-criteria scheduling with millisecond-level decision times. Moreover, the proposed migration mechanism reduces downtime by more than 50 % compared to vanilla Kubernetes, ensuring service continuity in mission-critical tasks. This work advances the vision of collaborative intelligence in IoT systems, bridging the gap between human judgment and automated orchestration for Industry 5.0 applications.

1. Introduction

The progressive and widespread integration of Artificial Intelligence (AI) and Machine Learning (ML) into industrial domains is driving a profound transformation of manufacturing and production ecosystems [1]. Supported by the large-scale deployment of Internet Of Things (IoT) devices and the maturation of edge computing technologies, modern industrial environments are increasingly leveraging distributed intelligence to enable real-time decision-making, enhance operational efficiency, and optimize resource utilization.

* Corresponding author.
E-mail addresses: riccardo.venanzi@unibo.it (R. Venanzi), lorenzo.colombi@unife.it (L. Colombi), davide.tazzioli@unibo.it (D. Tazzioli), simon.dahdal@unife.it (S. Dahdal), mauro.tortonesi@unife.it (M. Tortonesi), luca.foschini@unibo.it (L. Foschini).

https://doi.org/10.1016/j.iot.2025.101830
Received 2 May 2025; Received in revised form 5 September 2025; Accepted 13 November 2025
Available online 26 November 2025

At the core of this evolution lies the concept of Collective Intelligence (CI), where distributed contributions from humans, sensors, and AI agents integrate into a context-aware and adaptive system capable of solving complex tasks in dynamic and heterogeneous environments [2,3]. In CI, intelligence emerges through the decentralized aggregation of local knowledge, without requiring centralized control. A complementary notion is Collaborative Intelligence (CoI), in which humans and intelligent systems explicitly cooperate: humans contribute expertise, contextual insight, and ethical reasoning, while machines provide computational power, pattern recognition, and real-time adaptability [4,5]. This dual perspective-emergent CI on one side, cooperative CoI on the other-frames the broader convergence of human and machine capabilities.

In parallel, advances in machine learning (ML), and particularly the emergence of Large Language Models (LLMs), have made human-machine collaboration increasingly seamless. LLMs offer natural and intuitive interfaces between humans and information systems, lowering the barrier to interaction and enabling more fluid knowledge transfer. This technological shift has accelerated the practical realization of both CI and CoI paradigms, effectively merging human creativity and contextual awareness with machine efficiency and scalability into a unified form of distributed intelligence.

Together, these developments underpin the transition from Industry 4.0 to Industry 5.0. While Industry 4.0 focused primarily on automation and digitalization, Industry 5.0 emphasizes human-centricity, resilience, and sustainability [6,7]. Imagine smart factories where skilled operators and AI-powered systems work in synergy, not in isolation or opposition, ensuring that productivity gains and technological innovation remain aligned with human values, safety requirements, and environmental impact [8,9].

Despite the transformative potential of CI and Collaborative Intelligence, realizing these paradigms in modern Industrial IoT (IIoT) settings introduces substantial technical challenges. These include the orchestration of heterogeneous edge resources, real-time service migration under tight performance constraints, and the dynamic adaptation of AI models to evolving contexts [10]. While platforms like Kubernetes (K8s) provide strong foundations for container orchestration, they are insufficiently equipped to manage stateful, latency-sensitive applications in edge-to-cloud environments [11]. Specifically, K8s lacks real-time scheduling policies, contextual resource awareness, and human-in-the-loop orchestration features.

Originally designed for cloud-native applications in homogeneous and resource-rich data centers, Kubernetes (K8s) struggles to meet the stringent requirements of industrial edge scenarios [12]. Key challenges include the lack of real-time and proactive service migration mechanisms, limited awareness of network and hardware heterogeneity, inability to reason about context-specific metrics such as latency and reliability, and the absence of built-in support for dynamic, collaborative human-machine decision-making [13]. The default reactive scheduling policies of K8s, which prioritize simple resource utilization metrics like CPU and memory, are insufficient for managing mission-critical services such as anomaly detection and predictive maintenance in highly dynamic industrial environments [14,15].

AI-driven applications such as anomaly detection and predictive maintenance play a crucial role in industrial environments, but face challenges due to limited edge resources. Lightweight models provide faster inferences but often sacrifice accuracy, while more complex and bigger models achieve higher precision at the cost of straining computational capacity. This creates a need for adaptive, context-aware service management that balances performance and resource constraints.

To address the limitations of existing orchestration frameworks and service management in Industry 5.0 scenarios, our work introduces a novel architecture that integrates collective and collaborative intelligence principles into Kubernetes-based service management. The key contributions of this paper are summarized as follows:

- **Collective intelligence-driven orchestration:** We propose a novel service migration framework that extends Kubernetes with collective intelligence mechanisms, enabling adaptive, context-aware orchestration across heterogeneous edge and on-premise cloud nodes.
- **Collaborative intelligence-based Zoom-In functionality:** We introduce the Zoom-In feature, which empowers human operators to trigger dynamic upgrades of ML models, thereby combining human expertise and automated intelligence for anomaly detection and other AI-driven industrial tasks.
- **LLM-powered intent and policy engine:** We integrate a new component, LLM-Powered Intent and Policy Processor, which exploits large language models to transform natural language operator instructions into machine-readable policies, ensuring intuitive, explainable, and context-aware decision-making.
- **Proactive and intelligent stateful service migration:** We design and implement a novel migration mechanism within Kubernetes that minimizes downtime through hot/cold state management, enhancing service continuity in mission-critical scenarios.

Together, these components provide a scalable orchestration solution tailored for Industry 5.0 environments, enabling transparent, human-guided control and effective integration of decentralized intelligence and human-machine collaboration. Finally, we validate the effectiveness of the proposed framework through deployment in real industrial environments. A comprehensive experimental campaign was carried out across four dimensions: anomaly detection, operator intent translation, service scheduling, and service migration. The results highlight that specialized ML models can achieve F1-scores up to 1.0 in anomaly detection, lightweight LLMs reliably translate operator requests into machine-readable policies with about 72 % accuracy in 3 seconds, multi-criteria scheduling algorithms compute decisions in under 10 ms, and the proposed migration mechanism reduces service downtime by more than 50 % compared to vanilla Kubernetes. These findings confirm the practical feasibility of our approach in balancing computational constraints with operational requirements, while advancing the reliability, adaptability, and human-centricity envisioned by Industry 5.0.

The remainder of this paper is organized as follows. Section 2 reviews the background and related works on AI-driven orchestration, service migration, and collaborative intelligence in Industry 5.0. Section 3 presents the proposed collective intelligence-based service migration framework, while Section 4 details the Zoom-In functionality and its workflow in a real industrial use-case. Sec-

tion 5 discusses the experimental evaluation, highlighting the performance of anomaly detection, operator intent translation, service scheduling, and migration. Finally, Section 6 concludes the paper and outlines directions for future research.

2. Background and related works

The adoption of AI and ML in manufacturing contexts has grown substantially, thanks to the large amount of data collected from the large-scale deployment of industrial IoT devices and recent progress in the AI field. Another key enabler is represented by all the solutions developed to make IoT devices from different manufacturers collaborate and share information. A noteworthy example is the conceptual architecture presented in [16], which has the goal of improving devices' proactive collaboration regardless of the proprietary protocols developed by the manufacturers. On the other hand, AI-based solutions are used both to enable a more intelligent and efficient service management and to optimize the industrial processes, for example, employing anomaly detection models. Moreover, ML models-particularly Large Language Models (LLMs)-are increasingly employed to provide intuitive interfaces between humans and information systems. This pervasive adoption of AI and ML, combined with the seamless integration of human and machine capabilities, has given rise to what is often referred to as Collective and Collaborative Intelligence. The result of these kinds of distributed intelligence is the merging of human and machine capabilities to form a unified global brain capable of tackling complex problems. The proliferation of IoT devices has advanced this vision, enabling large-scale coordination across hundreds or thousands of nodes to pursue shared objectives. At the same time, increasing computational power at the edge allows intelligence to be deployed closer to the data source, supporting soft real-time inference even in mission-critical cyber-physical systems [17–19]. Nonetheless, this fast shift toward the edge presents many challenges, stemming from the heterogeneity of devices with diverse software stacks, compute architectures, and resource capacities [20]. These challenges include, for example, security governance and assurance [21] and AI trustworthiness [22].

Besides the AI and ML applications in the industrial domain, Anomaly Detection, in particular, has consistently been one of the most sought-after industrial applications [23]. This is usually performed on time series data [24], which plays a crucial role in ensuring operational efficiency and safety across a range of applications. The inherent unpredictability of such anomalies introduces substantial challenges. Unlike standard predictive modeling, where the target outcomes are predefined and well understood, anomalies often represent entirely novel behaviors or patterns that are neither previously encountered nor labeled. Consequently, effective anomaly detection necessitates algorithms that are both robust and capable of generalizing well to identify deviations beyond the scope of historical training data.

Time series anomaly detection methods are generally divided into three main categories: distance-based, density-based, and prediction-based [25]. Distance-based approaches identify anomalies directly from the raw time series by applying distance or similarity measures, without requiring explicit modeling of temporal structure. Density-based methods, in contrast, do not treat the time series merely as numerical values but instead embed them into richer representations, such as distributions, graphs, or trees, to uncover irregular patterns in their structural or statistical properties. Finally, prediction-based techniques seek to model normal behavior by training on historical time series or subsequences; anomalies are flagged when observed values deviate significantly from the predicted ones. Within the prediction-based family, two subcategories dominate industrial practice. Forecasting-based models, often realized with Long Short-Term Memory networks (LSTMs) or Recurrent Neural Networks (RNNs), predict future values and detect anomalies when deviations exceed a threshold. Reconstruction-based methods, typically implemented with Autoencoders (AEs) or Convolutional Autoencoders (CAEs), reconstruct the input signal and identify anomalies through unusually high reconstruction errors [26,27]. More recently, encoding-based approaches have been introduced: here, an autoencoder extracts a compact vector representation of the time series, which is then processed by classical distance-based anomaly detection algorithms [28].

As a consequence of the wide range of possibilities and the challenges inherent in managing and operating ML-based services, such as sustaining accuracy over time, detecting out-of-distribution inputs, and ensuring overall trustworthiness, it is essential to adopt an infrastructure that supports both service management and Machine Learning Operations (MLOps) functionalities. To this end, K8s has emerged as the de facto standard for service orchestration and management and is increasingly employed in complex, multi-cluster edge-to-cloud compute continuum scenarios, particularly within the context of Industry 5.0. A representative example can be found in [29], where the authors propose an automated service orchestration platform that integrates user-owned devices as end nodes within the edge-to-cloud compute continuum and enables service deployment on them.

Usually, these settings are made of one main cloud-based K8s cluster and many edge clusters, located near the production plant, with limited computational resources [30]. However, K8s by design is not intended for comprehensive infrastructure management, as its primary role is to manage the lifecycle of services. Its default scheduler focuses on optimizing resource allocation based on CPU and memory availability, but it lacks real-time insights into factors like network latency, bandwidth variability, and dynamic workload changes-elements that are crucial in industrial environments. Traditional K8s scheduling mechanisms are reactive and do not support built-in workload migration. This means workloads are only moved after performance issues arise. In contrast, Industry 5.0 demands proactive migration strategies to avoid bottlenecks and ensure minimal downtime [31]. Without AI-driven predictive analytics, K8s struggles to adapt in real time to the demands of evolving industrial workloads.

In the literature could be found several works about service migration employing general-purpose checkpointing methods, such as CRIU without tailoring to industrial needs [32], while others, like Koziolek's approach, integrate with K8s for stateful migration of industrial control containers using synchronized clocks and OPC-UA [33]. Another noteworthy example can be found in [34], where the solution presented uniquely targets ML workloads in Digital Twin contexts. However, these methods often overlook ML-specific requirements and dynamic multi-cluster cloud-edge environments central to Industry 5.0. A comprehensive solution should incorpo-

rate automation mechanisms, along with both AI-enabled and AI-enabling service orchestration, to ensure effective coordination and management of functionalities at the platform level.

Trying to solve these challenges, both researchers and businesses started applying Zero-touch network and Service Management (ZSM) techniques to Industry 5.0 environments, although they were previously adopted especially for large clusters in the telecommunication field. ZSM is a framework aimed at achieving fully automated, zero-touch management of networks and services, independent of vendor-specific implementations. Its architecture supports flexible, vendor-neutral management capabilities, aligning with industry trends that favour adaptable, dynamic management systems over traditional, rigid alternatives [35]. These features make this framework valuable in a smart manufacturing environment, too.

The ZSM framework adopts a modular and loosely coupled architecture, enabling independent scaling and deployment of various management modules and services. A core attribute of ZSM is its closed-loop automation capability, which allows networks to autonomously meet and sustain operational objectives without human input. This functional decoupling from data storage and processing fosters improved portability, reusability, and interoperability across heterogeneous vendor environments and infrastructures [36].

Within ZSM, management functions are distributed across domains that abstract resource complexities from the perspective of service users. Automating management processes enhances the flexibility and efficiency of service delivery, reducing Operational Expenditures (OPEX) by incorporating self-managing functionalities such as self-configuration, self-healing, self-optimization, and self-protection [35]. The end-to-end (E2E) service management function operates across these domains to ensure seamless coordination. Additionally, ZSM employs intent-based interfaces that enable high-level abstraction of desired network behaviours and automatic translation of service requirements into actions.

Intent-based management is pivotal in enabling automation within the ZSM paradigm [37]. It simplifies the automation of network and service management by allowing users to specify high-level, outcome-focused goals rather than concrete instructions [38]. This allows systems to determine optimal strategies and adapt dynamically and autonomously. To ensure broad compatibility, intent statements must be infrastructure-agnostic. Unlike conventional systems, intent conflicts are managed at runtime, necessitating dynamic adaptation. Furthermore, intent statements serve as a universal format to express expectations across various domains and management layers [37]. These functionalities make it possible for non-technical operators in Industry 5.0 to interact with the IT system, adapting it to changing situations.

A formal definition of intent is sourced from the “TM Forum IG1230” [37] exploratory report: “Intent is the formal specification of the expectations, including requirements, goals, and constraints, given to a technical system.” According to the same source, intent must be declarative-focused solely on the desired outcomes without prescribing implementation details-and must be both machine-processable and human-readable to facilitate clear, interoperable communication across system components.

Against this backdrop, the emergence of LLMs within the Natural Language Processing (NLP) field has sparked interest in leveraging them to convert natural language-based intent statements into structured formats [39], potentially contributing to their standardization in network and service management domains [35]. Once intent is translated into machine-readable policies-whether individual rules or rule sets [40], or expressed in declarative languages such as Prolog [41]-symbolic AI can be employed to reason over these policies in conjunction with a knowledge base that encapsulates system states (e.g., resource usage, hardware capabilities, cost, and latency) to support decision-making.

Summarizing, the literature review highlights several open challenges that remain insufficiently addressed, and our proposed architecture is designed to directly tackle them. First, while anomaly detection methods are well explored, they seldom account for deployment constraints in heterogeneous, resource-limited edge-to-cloud environments. Our architecture incorporates adaptive, lightweight ML deployment strategies that ensure robustness under such constraints. Second, orchestration platforms such as Kubernetes are widely adopted but lack proactive, ML-aware mechanisms to guarantee reliability and low latency in dynamic industrial workloads. We extend this by integrating predictive analytics and workload-aware scheduling to enable proactive orchestration. Third, frameworks like Zero-touch Service Management (ZSM) provide closed-loop automation and intent-based control, yet they remain untailored to ML-specific requirements and multi-cluster compute continua. Our design adapts ZSM principles to explicitly support ML life cycle management in distributed industrial contexts. Finally, the integration of intent-based management with natural language interfaces, supported by LLMs, is largely unexplored in practice. Our approach leverages LLMs to bridge human-machine interaction, enabling intuitive, intent-driven orchestration. Taken together, these contributions demonstrate how the proposed architecture addresses the key shortcomings identified in the literature, offering an adaptive, ML-centric, and intent-driven solution for heterogeneous industrial environments.

3. Collective intelligence-based service migration framework

The proposed framework aims to orchestrate intelligent, adaptive, and context-aware service migration in industrial IoT environments through a dual-paradigm approach combining **collective** and **collaborative intelligence**. It is specifically tailored for Industry 5.0 scenarios where human-machine symbiosis is paramount, leveraging K8s as its orchestration substrate while significantly enhancing its capabilities.

The architecture operationalizes **collective intelligence** by enabling decentralized edge devices to autonomously share real-time metrics (e.g., latency, resource usage) that are aggregated into emergent migration policies. Simultaneously, it embodies **collaborative intelligence** through human oversight - operators refine decisions via natural language interfaces and validate critical actions like the Zoom-In functionality. This blended approach ensures both system-wide optimization (via collective automation) and human-centric adaptability (via collaborative control).

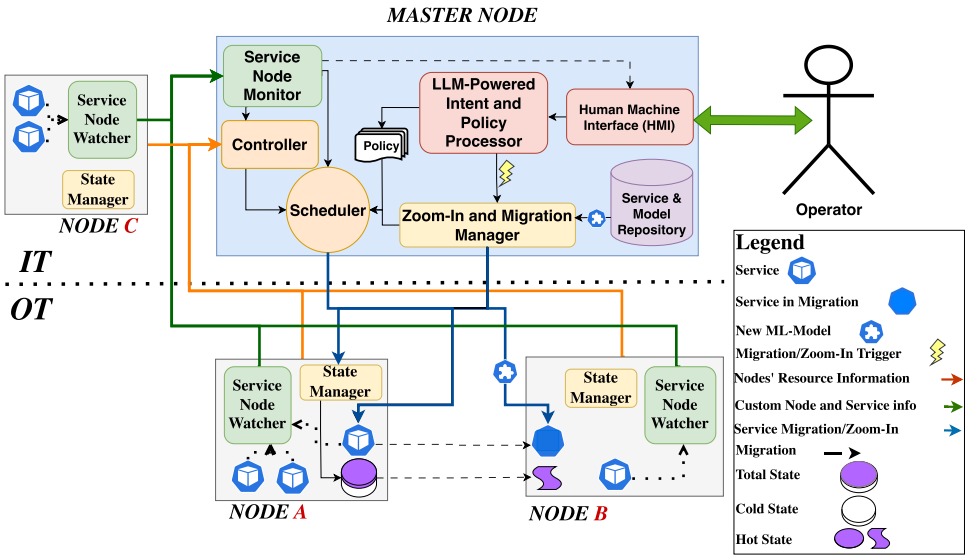


Fig. 1. Collective intelligence-based service migration framework architecture.

The framework introduces two key innovations:

- An advanced stateful service migration mechanism that extends traditional K8s architecture with collective intelligence-driven orchestration
- The Zoom-In functionality leverages MLOps and Collaborative/Collective Intelligence principles, using dynamically selected ML models to adapt to the required level of analysis through collaborative human-machine decision-making.

3.1. Framework design and architecture

We place our proposed architecture in a typical Industry 5.0 environment composed of a master node running the core of the proposed framework, typically deployed on the IT level, and several edge nodes. The system's edge nodes are deployed near industrial assets and are responsible for running real-time services such as anomaly detection or predictive maintenance. These nodes are typically constrained in computational resources and memory, necessitating intelligent load balancing and dynamic service relocation when higher demands arise. The control logic of the framework resides within a central master node that coordinates the behavior of all connected nodes.

The proposed framework, depicted in Fig. 1, orchestrates adaptive and context-aware service migration in industrial IoT environments, emphasizing the principles of Industry 5.0 combined with collective intelligence. It operates as a K8s-based system capable of dynamically managing containerized stateful services at both the IT and OT levels. In addition, the proposed framework provides collaborative intelligence-based decision-making functionality through an LLM-driven engine that combines human know-how and expertise with advanced AI algorithms. Fig. 1 depicts the proposed framework and its components. The color code used for the components in the figure is feature-driven, and it groups the components by the framework's functionalities. We have depicted in green the entities responsible for providing service awareness by monitoring custom metrics (Service Node Monitor and Service Node Watchers). Following the same path, yellow is the color assigned to components that handle the Stateful Service Migration and the Novel Zoom-In Functionality (Zoom-In and Migration Manager, and State Managers). The components responsible for managing the Human-in-the-Loop feature and their interaction with the framework are colored red. Orange is the color assigned to K8s-native components we empowered in the proposed framework. Finally, we depicted the component responsible for storage in purple (Service and Model Repository).

The master node consists of several novel interacting components and enhanced K8s modules. The ***Service Node Monitor*** embodies collective intelligence by continuously collecting and aggregating custom metrics about each node, like latency, and special data from the running services, creating a system-wide awareness that informs migration decisions. Custom metrics are important for heterogeneous environments, such as industrial ones, in which the demands and the needs can drastically vary depending on the applications, use cases, type of deployments, or even the production site. A pivotal custom metric to collect might be the latency of the cluster node; it might be paramount in case of real-time applications, i.e., process or product anomaly detection, which has to be deployed on the node with the lowest latency possible. The other type of data the Service Node Monitor collects is the special app data. Those data are application-specific and can employ alerts or particular data that the operator must check.

The ***Controller*** is a K8s module responsible for collecting node metrics such as CPU, memory consumption, and application health. Then, it analyzes them in order to enable a fully contextualized and aware orchestration.

The ***Scheduler*** and ***Zoom-In and Migration Manager*** jointly determine when and where services should be migrated or upgraded and enable the Zoom-In functionality. More in detail, the Scheduler is an enhanced K8s module capable of providing stateful service migration and Zoom-In features, both detailed in Section 3.3 and in Section 4, respectively. The Scheduler is a custom version of the K8s vanilla one; it is specifically developed to add support for stateful service migration and, consequently, to the Zoom-In feature. It receives the custom metrics from the Service Node Monitor, the nodes' resource consumption from the Controller, and the policies to select the best node to migrate the service and when to migrate it. The Zoom-In and Migration Manager is responsible for coordinating the service migration and the Zoom-In functionality. It bridges both collective and collaborative intelligence by implementing an optimized resource allocation service migration and a human-triggered Zoom-In action. It syncs with the Scheduler to run the service migration and control the State Manager on the involved nodes, and receives the Zoom-In trigger to start the procedure and pilots the scheduler to trigger the service shift.

Additionally, the component ***Service & Model Repository*** stores containerized versions of ML models and services. The framework leverages this repository to power the Zoom-In functionality, dynamically selecting and swapping models to adapt the analysis to the specific requirements of collaborative human-machine decision-making. For instance, lightweight and fast models can be used for real-time analysis on edge devices, while more complex models can be deployed on resource-rich nodes for deeper insights. This functionality is described in detail in Section 4.

The ***State Manager*** is deployed on the nodes, and it is piloted by the Zoom-In and Migration Manager. It is responsible for the state transfer of the service in-migration or during the Zoom-In functionality.

The key enabler of intelligence in this architecture is the ***LLM-Powered Intent and Policy Processor***, which interprets operator-defined policies to control the migration and to trigger the Zoom-In functionality, in an intent-driven fashion. More in detail, this module interacts with the field operator to extract the policies to select the destination node that best suits the specific industrial case. In addition, through the ***Human-Machine Interface (HMI)***, special application data from running services, i.e., anomaly detection application, are shown to the operator who can decide according to their expertise or know-how whether to trigger the Zoom-In functionality or not. In case of activation, LLM-Driven Decision and Policy Engine interprets the operator commands and instructions and triggers the procedure. Finally, the LLM-Driven Decision and Policy Engine can also select, according to the operator's instruction, which ML model suits better with the application-specific scopes.

3.2. Human-centric component: LLM-powered intent and policy processor

This component, illustrated in Fig. 2, has the goal of enabling human-machine Collaborative Intelligence, translating a user's intent, which comes from the operator through the HMI and initially expressed in natural language, into a machine-readable policy, offering both flexibility and ease of interaction. In this way, based on their domain expertise or operational know-how, the operator can then decide whether a service reconfiguration is needed. For example, when an anomaly detection model detects an anomalous behaviour, a more precise model could be used to perform root cause analysis. This could require a system reconfiguration, because the node where the model is actually deployed could not have enough resources to run the second, heavier model. We refer to this model change and system reconfiguration as the "Zoom-in Functionality", which is better explained in Section 4.

The Intent Ingestion Component, powered by an LLM, interprets operator commands and natural language instructions, translating them into structured, machine-readable actions that can be executed across the system. This module evaluates whether the statement is sufficiently clear and well-defined or if it needs further clarification. If any ambiguity is detected, the system prompts the user to provide the missing or unclear details to ensure accurate interpretation. Notably, the ingestion and translation stages could, in principle, be implemented by a single LLM capable of both interpreting and converting the intent.

Once the intent is considered well-specified, the intent translator, also an LLM-powered component, converts it into a machine-readable format, for example, JSON or YAML, without compromising human readability to preserve interpretability. This intermediate representation also serves to decouple the translation stage from downstream processes, allowing for easier substitution or modification of components within the pipeline.

Once the procedure is triggered, the LLM-Driven Decision and Policy Engine coordinates the downstream workflow, determining which resources, services, or subsystems need to be engaged. It also supports adaptive model selection: according to the operator's instructions and the application-specific scope, the engine can evaluate available ML models and select the most appropriate one for the task, other than choose the amount of the state that has to be migrated together with the model. The choice may be based on multiple criteria such as model accuracy, computational efficiency, latency requirements, or robustness to edge cases. Noteworthy, the actual model and state dimension choice is not defined directly by the LLM; instead, it defines a series of policies, including constraints, which are later used to select the model that will be deployed.

Following translation, a checker module examines the machine-readable policy file to ensure it is syntactically correct, specifically assessing the file format and the presence of all the required attributes, and possibly that all original intent information has been accurately preserved. For example, we can check if the output file has a correct JSON syntax and if it is compliant with a predefined schema. If the validation fails, the translation module is prompted to retry. After a certain number of unsuccessful attempts, the user is notified of the persistent issue.

Upon successful validation, the JSON file is passed to an actuator module that transforms the intent into a low-level policy that is subsequently used by the Migration Manager, which uses them to perform decision-making, specifically determining the nodes where the service has to migrate, the new model, and the dimension (e.g. how many data samples) of state that has to be moved to the new location. In this way, the LLM not only functions as an intelligent interpreter of human intent but also as a dynamic decision-making layer that aligns human expertise, system capabilities, and policy-driven constraints into a cohesive operational process.

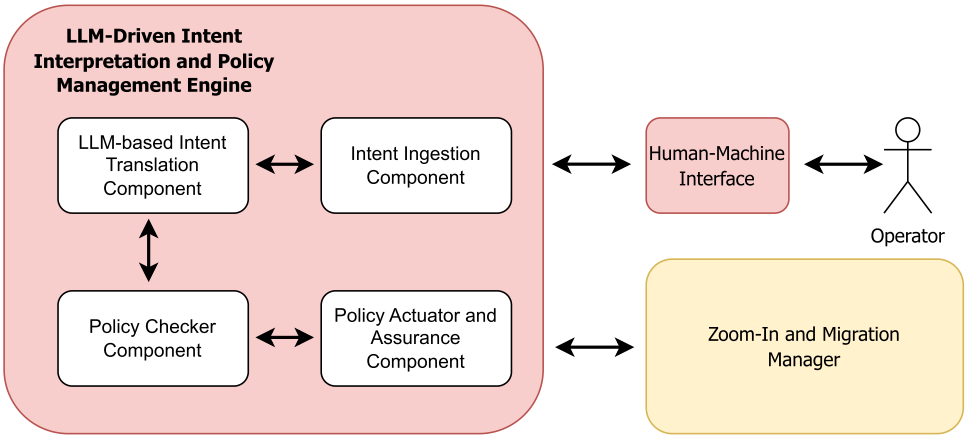


Fig. 2. LLM-driven intent interpretation and policy management engine architecture.

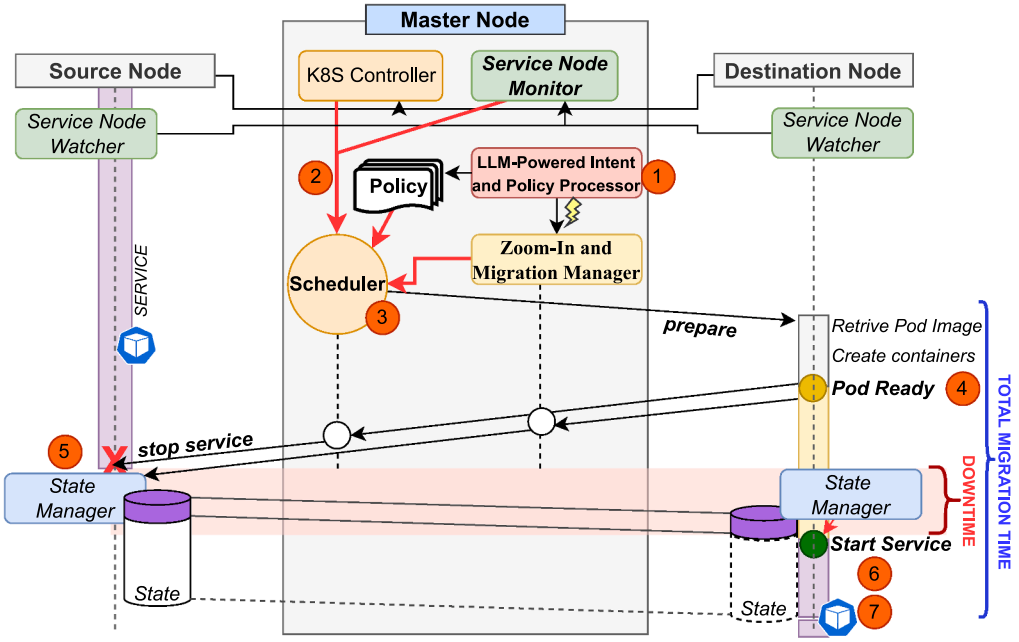


Fig. 3. The migration mechanism.

Finally, an intent assurance module monitors the system by continuously collecting metrics from the network's nodes. This monitoring ensures that the implementation aligns with the specified intent and provides feedback to earlier stages in the pipeline, enabling corrective actions if discrepancies are detected.

3.3. Collaborative intelligence driven migration mechanism

In this section, we describe in detail the novel platform-level mechanism for the dynamic placement and migration of application services. The mechanism is designed to meet the fundamental requirements of modern industrial environments, particularly high availability and operational safety. In practice, this translates into solutions that aim to maximize service availability during migration. The proposed approach handles migration entirely at the infrastructure level, transparently to the migrated services; as a consequence, live migration is not provided, since no application-level synchronization is implemented. The application state is efficiently managed to prevent inconsistencies by design, avoiding the parallel execution of multiple service instances. Previous analyses and prototypes we developed form the basis of this design and are documented in [31] and [42].

The migration mechanism we introduce is illustrated in Fig. 3, and involves components previously described in Section 3.1. Collective intelligence emerges from the combination of human-defined high-level policies and real-time cluster metrics. On one side, policies are generated through the LLM-Powered Intent and Policy Processor component, based on high-level human instructions. On

the other hand, these policies are enriched with metrics collected from the nodes through the K8s default controller and the newly introduced Service Node Monitor for context-specific observation (1). The scheduler leverages this joint input to trigger migrations at the most optimal moment (3). Likewise, the *Zoom-In and Migration Manager* can initiate a service migration (2) in contexts aligned with the Collaborative intelligence model. An example is the zoom-in functionality, detailed in Section 4, where human interaction is required during the infrastructure management process.

A key strength of our solution is the preparatory phase introduced in the migration mechanism, which configures the destination node before interrupting the currently active service. This stage includes initializing the pod environment, pulling the corresponding service image, and instantiating the required containers, stopping just short of actual activation. Notably, this phase runs in parallel with the service execution on the source node, which continues to provide the service during the migration operations. This proactive migration approach accelerates the activation of the new service, as it can be immediately reactivated on the destination node following deactivation on the source node.

The prepared container on the target node is labeled by the K8s controller manager in the newly defined *ready-state* once the preparatory stage has been completed (4). At this point, migration begins: the running instance on the source node is deactivated (5), the state is transferred, and the ready container on the destination node is activated.

Moving state is a delicate and potentially costly operation in terms of time and network consumption. To further reduce service downtime, we introduce the State Manager component on each node, as described in Section 3.1. This component identifies the portion of data required for the immediate reactivation of the application on the destination node and transfers it with priority. This critical portion is referred to as the *hot state*. Defining the hot state is highly dependent on the service's application logic, unlike the other migration operations, which are designed to operate entirely at the infrastructure level. For this reason, state partitioning is delegated to the external State Manager, which is configured with high-level policies by the LLM-Powered Intent and Policy Processor through the Zoom-In and Migration Manager. The State Manager is activated only during the migration process, as it is not intended to function as a backup system; therefore, no periodic state analyses are performed outside migration operations. Once the transfer of the hot state is complete, the previously prepared container is immediately reactivated, making the service available again (6). The remaining portion of the state, referred to as the *cold state*, is then transferred in the background after the service has resumed, introducing no further unavailability (7).

Managing hot and cold states is a sensitive aspect of the migration process, since it directly determines how much downtime an application will experience. Identifying which portion of the application's data is essential for immediate operation (hot state) versus what can be deferred (cold state) is not a fixed rule but instead depends heavily on the model being executed and the task it is performing. For example, an anomaly detection model may only require a short time window of recent data to continue functioning correctly, whereas a recommendation system might need a larger history of user interactions to maintain accuracy. An example is presented in [42]. In this study, we measured a hot state of 1 MB for a lightweight anomaly (e.g., an isolation forest model). This value corresponded to the average working window size of the Anomaly Detection application, which was configured to analyze the last three minutes of data collected by the machine. The preliminary analysis highlighted the operations contributing to the total migration time, with particular focus on a stateful transfer involving a dataset of 500 MB.

Because of this variability, the Zoom-In and Migration Manager component on the master node is responsible for dynamically configuring the state manager on the nodes during migration. Guided by high-level policies from the LLM-powered Intent and Policy Processor and a series of human-defined empirical rules, it decides which data must be transferred immediately to preserve service continuity and which data can be migrated later in the background without affecting the application's performance.

For instance, in the use case of a zoom-in functionality targeting a portion of data, the hot state will be defined by the specific data to be analyzed, possibly supplemented later in the background by additional data. Alternatively, some policies may prioritize minimizing network consumption over reducing downtime, thereby applying a different approach to state management. This strategy allows for great versatility and system stability; implementation mechanisms remain separated from high-level policies, which can thus better represent the specific needs of the application context.

4. Zoom-In functionality

The Zoom-In functionality is a novel key feature provided by the proposed framework. It embodies the collaborative decision-making and dynamic adaptability necessary for intelligent industrial IoT applications. This mechanism, originally sketched in [43], enables the system to upgrade the intelligence of a deployed service in response to potential anomalies, early warning signs of equipment degradation, or particular application-specific data.

4.1. A collaborative intelligence-driven functionality

The Zoom-In functionality exemplifies a practical implementation of collaborative intelligence, where the strengths of human operators and intelligent computational agents are brought together to enhance system behavior within the Industry 5.0 environments. More specifically, rather than delegating all decisions to automation, this framework fosters a CI in which the system and the human operator co-participate in the decision-making loop. Each brings distinct and complementary capabilities: the operator contributes domain knowledge, contextual awareness, expertise, know-how, and even ethical judgment, while the system offers real-time monitoring, rapid inference, and access to a global state view across distributed nodes.

The Zoom-In functionality is inherently general-purpose and applicable across a wide range of domains, including smart cities, healthcare, defense, and disaster recovery. It is particularly well-suited to advanced industrial scenarios. In such contexts, the in-

tegration of human operators within the decision-making loop fosters a novel form of human-in-the-loop collaborative intelligence, ensuring that critical decisions remain context-aware and are not left to blind automation. This approach enhances both the reliability and explainability of AI-driven systems in complex modern industrial scenarios, high-stakes environments.

In today's industrial landscape, AI-driven applications have become a foundational component of production environments, particularly in use cases such as anomaly detection and predictive maintenance. These applications are often deployed on constrained edge nodes due to their proximity to machines and processes, enabling low-latency monitoring and immediate feedback. However, given the limitations in computational and memory resources on such nodes, these deployments typically rely on lightweight models, although efficient, these models may lack the depth required to recognize many atypical anomalies or may present uncertainties. Therefore, sometimes could be useful to switch models to a more powerful one to better investigate the possible anomaly. When this is detected, the event is reported to the master node, which in turn notifies the human operator through the HMI. It is at this point that CI becomes fully operational: the system does not autonomously escalate the situation blindly or execute an operation autonomously, but instead defers to the human operator, who, through their expertise and contextual understanding, evaluates whether the anomaly warrants deeper investigation. If the operator deems the anomaly significant, a more sophisticated model, with higher precision and larger scope, must be deployed to perform high-fidelity analysis. The deployment of such models, however, introduces a substantial increase in computational load, often exceeding the capabilities of the originating edge node. Attempting to run these models directly on constrained hardware could lead to performance degradation or system failure due to CPU and memory saturation. To address this, the Zoom-In functionality leverages MLOps practices in conjunction with runtime orchestration policies to transparently migrate the stateful anomaly detection service to a more capable node (or with more available resources).

This migration is not merely a transfer of computation, but a strategic adaptation of system architecture, driven by a combination of operator intent, system telemetry, and policy-driven decision making. The original service, along with its runtime state, is transferred to a destination node selected based on available resources and latency requirements. Crucially, the upgraded, more powerful anomaly detection model is deployed directly on the new node, thereby avoiding overload on the original, resource-constrained host.

In doing so, the Zoom-In functionality operationalizes the core principles of CI: it allows the system to scale its analytical capabilities only when justified by human insight, and to do so in a way that is context-sensitive, efficient, and aligned with the physical limitations of the deployment environment. This intelligent interplay between human judgment and system adaptability ensures that industrial AI deployments remain both robust and responsive, embodying the vision of Industry 5.0 as a human-centered, resilient, and intelligent production paradigm.

4.2. Zoom-In workflow

The Zoom-In process is an implementation of collective intelligence in industrial environments, and this subsection is meant to unravel its workflow. We implemented this functionality in a real modern industrial deployment and depict its operational workflow in Fig. 4. The figure shows only the component of the architecture involved in the Zoom-In procedure.

The whole workflow is composed of eight main steps described as follows:

1. **Data Collection and Processing:** The anomaly detection application running on edge node A collects data directly from the industrial machinery on the shop floor and analyzes it through a lightweight model;
2. **Anomaly Detected:** As soon as the application detects a possible anomaly during the production process, it publishes the detected information to the Service Node Watcher that sends it immediately to the Service Node Monitor deployed in the Master Node;
3. **Anomaly Reported** The Service Node Monitor receives the reported anomaly and alerts the Human Operator through the HMI;
4. **Human Inference:** The reported possible anomaly is analyzed by the human operator, who evaluates it and decides if the anomaly deserves a more accurate investigation. The human operator, according to their expertise, know-how, context, and application-specific information, establishes if the reported possible anomaly is a true or false positive and if it has to be further investigated. If yes, through the natural language, the human operator starts the Zoom-In procedures;
5. **Zoom-In Trigger:** The LLM-Powered Intent and Policy Processor translates the operator instruction, selects the advanced ML model to adopt for the advanced detection, and triggers the Zoom-In and Migration Manager;
6. **Zoom-In and Service Migration:** The Zoom-In and Migration Manager receives the trigger for the Zoom-In functionality and fetches the new advanced ML model from the Service & Model Repository, and informs the Scheduler to start the migration procedure;
7. **Migration and New Model Deployment:** The Zoom-In and Migration Manager and the Scheduler execute the Zoom-In functionality by selecting the destination node and configuring the State Manager by defining the amount the state to transfer immediately (hot state) and the state to move in the background. In parallel, the new advanced ML model is deployed on the destination node;
8. **Service Restoration:** The anomaly detection service is migrated and restored with the new advanced ML model. The application resumes working and analyzes with a larger scope and with a more accurate model the production process. The application can confirm the anomaly or detect and report new anomalies through the Service node Watcher, and the procedure can start again. If the anomaly is confirmed, or a new, more severe anomaly is detected, the human operator is informed, and they can decide on a proper counteraction, such as to interrupt the process or to change production parameters.

The described procedure focuses exclusively on the case where the Zoom-In functionality is triggered. If, however, the operator determines that the reported anomaly is a false positive or not significant, they may choose to take no action, allowing the system to continue normal operations uninterrupted.

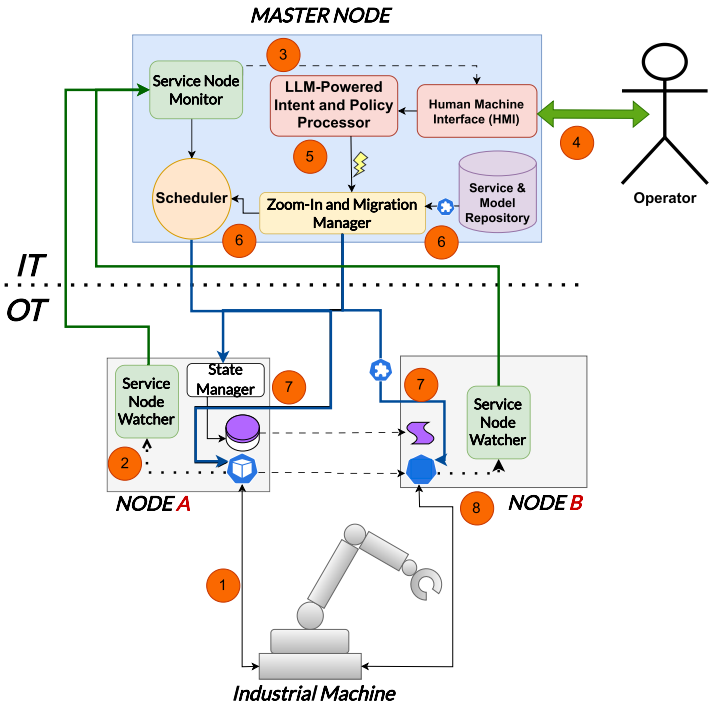
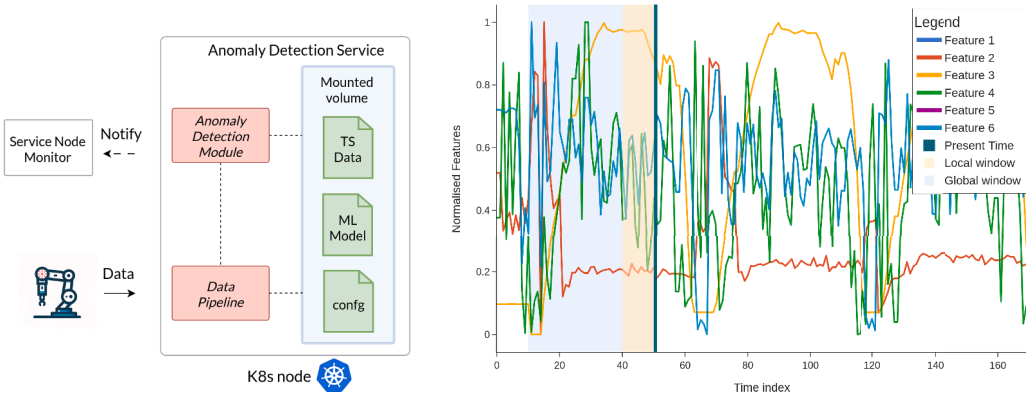


Fig. 4. Zoom-In functionality workflow.



(a) Application architecture (b) Time series data collected from the industrial machine

Fig. 5. End-to-end anomaly-detection prototype: (a) Application architecture and (b) Example multivariate time-series collected from the industrial machines.

4.3. Use case: Collaborative anomaly detection in industry 5.0

In this section, we introduce and motivate the anomaly detection use case in industrial processes, which gives practical value to the proposed mechanisms and solutions. Beyond its significant relevance and wide industrial applicability, this use case enables a comprehensive evaluation of our architectural proposal, covering both fast real-time analyses and more in-depth analyses leveraging the proposed Zoom-In functionality. This use case serves as a guiding thread for the tests and experimental results presented in Section 5.

Within the paradigm of Industry 5.0, advanced Anomaly Detection systems are emerging as critical enablers for manufacturing machinery. By identifying initial faults in spindles, drives, hydraulics, or control electronics before they escalate, such systems minimize unplanned downtime, lengthen asset lifetime, and stabilize production schedules, outcomes directly aligned with the economic and environmental goals of modern factories. Early recognition of abnormal thermal, vibrational, or power-consumption patterns further enhances shop-floor safety and regulatory compliance. Hence, reliable anomaly detection is not a mere add-on, but it is a foundational capability for factories that aspire to be productive, green, and resilient.

Conceptually, Anomaly Detection in manufacturing diverges from conventional multi-class classification because anomalous events are, by definition, rare, diverse, and often never-before-seen. Whereas a classifier presupposes a finite, labeled taxonomy of fault categories, an Anomaly Detection model first constructs a compact representation of "normal" machine behavior, frequently through various Machine Learning and Deep Learning algorithms such as one-class, unsupervised, or self-supervised learning techniques such as auto-encoders, or density estimators - and then flags observations that violate this learned pattern.

Traditional one-size-fits-all strategies are inadequate because the statistical signature of an incipient defect depends strongly on both its physical origin and its temporal footprint. Slowly evolving phenomena such as tool wear, thermal drift in spindle bearings, or gradual hydraulic leakage tend to manifest as low-frequency trends that span thousands of control cycles; they are captured most effectively by models that incorporate long receptive fields - e.g., dilated temporal convolution encoders with large context windows that subtle deviations from the seasonal baseline are not mistaken for noise. By contrast, sudden events such as impact chatter, misfires in servo amplifiers into controller area networks, generate short-lived spikes whose diagnostic content is concentrated within a few milliseconds. In these cases, models with high temporal resolution and minimal latency, such as small models operating on small sliding windows, excel because they avoid the smoothing that would dilute the evidence of the anomaly. These observations motivate the design of a unified anomaly-detection application that can dynamically select, weight, or compose multiple detection models as a function of live context supplied by the monitoring stack.

In Fig. 5, we present an anomaly detection application. Fig. 5(a) illustrates the application's architecture, which is deployed as a K8s pod that exposes two cooperating microservices. The *Data Pipeline*, which ingests the raw sensor stream emitted by the service-node monitor of the manufacturing machine, pre-processes it as needed, and writes it to a mounted time-series volume that is shared with the *Anomaly Detection Module*. The latter periodically reads the freshly accumulated data, loads a pre-trained specific ML model together with the corresponding configuration files from the same volume, and computes anomaly scores in near real time. Detected anomalies trigger asynchronous notifications that are sent back to the shop-floor monitoring layer, enabling immediate operator intervention or automated mitigation. All elements, sensor data, models, and configuration artifacts remain co-located inside the pod, which simplifies version control and allows hot-swapping of models without restarting the service.

5. Tests and experimental results

This section presents the experimental evaluation of the main components of the proposed architecture and their integration into a unified framework. The goal of this evaluation is twofold: first, to demonstrate the effectiveness of each sub-functionality when considered in isolation, and second, to highlight how their combined operation enables the architecture to meet the stringent requirements of Industry 5.0 environments, such as reliability, adaptability, low latency, and seamless human-machine collaboration. To this end, we structure the evaluation around four core pillars of the system. We begin with Anomaly Detection, where we examine the ability of different ML models to capture diverse types of irregularities in time series data, emphasizing the importance of model specialization for industrial use cases. Next, we turn to Operator Intent Translation, a critical component for enabling intuitive human-system interaction. Here, we assess the accuracy and efficiency of large language models in converting natural language intents into machine-readable policies, an ability that directly impacts the usability and adaptability of the entire framework. The third component, Service Scheduling, is evaluated through Multi-Criteria Decision-Making (MCDM) methods, where we test and compare different algorithms in terms of computational performance and scalability in ranking deployment options under multiple constraints. Finally, we analyze Service Migration, focusing on performance metrics such as migration time, service downtime, and resource consumption, to validate how the proposed mechanisms ensure continuity and high availability in dynamic edge-to-cloud environments.

5.1. Anomaly detection

To provide a concrete assessment of our Anomaly Detection application, we present a set of controlled experiments that highlight how specific anomaly classes demand correspondingly specialized ML models.

5.1.1. Dataset preparation

We collected three distinct multivariate time series datasets from sensors embedded in an industrial hobbing machine used to manufacture gears. The datasets consist of samples with durations of 30 seconds, 100 seconds, and 180 seconds, respectively. This range of sample lengths reflects realistic variations in operating conditions, where anomalies may or may not appear depending on the scenario. The inclusion of different durations allows for a comprehensive evaluation of anomaly detection models across various temporal resolutions. It also enables testing the robustness of data preprocessing and feature extraction methods on non-uniform time segments.

Each multivariate time series (MTS) dataset consisted of six features, as illustrated in Fig. 5(b). These features capture the load applied to various movable components of the system, including the piece holder table, the spindle, and the radial, tangential, and axial slides responsible for motion along different axes. Each dataset was partitioned into training and test subsets using a 90/10 split. To thoroughly evaluate the performance of anomaly detection models under realistic industrial conditions, we created two augmented versions of each original test subset. These augmented test sets were generated by injecting synthetic anomalies and extraneous noise into the MTS data. Two types of synthetic anomalies were introduced, guided by domain expert input to reflect common real-world disturbances in industrial machinery. The first type-step anomalies involved abrupt and sustained level shifts in one or more signal channels. The second type of periodic spike anomalies was created by superimposing sequences of transient spikes, with amplitudes

sampled from a three-component Gaussian Mixture Model. These anomalies simulate typical mechanical disruptions, such as sudden load changes and periodic mechanical chatter, frequently encountered in industrial environments. To assess the robustness and precision of the detection models, we also injected random extraneous noise into a subset representing 10 % of the original data. This included single-point impulse noise and salt-and-pepper noise, simulating transient, non-informative disturbances unrelated to actual mechanical faults. This noise scenario was intentionally designed to test the models' ability to distinguish between genuine anomalies and irrelevant, sporadic perturbations.

As a result of this augmentation, two distinct test sets were established for each MTS dataset. Test sets labeled "A" contain only synthetic anomalies introduced into four selected features. Test sets labeled "$A + N$" include both synthetic anomalies and extraneous noise within the same four features. This systematic augmentation strategy enables a rigorous evaluation of anomaly detection algorithms, assessing their capacity to accurately identify true anomalies in complex and noisy industrial data settings.

5.1.2. Experimental results

In Table 1, we observe a broad range of detection performance across the six test sets and anomaly detection methods. The F1-score, precision, and recall vary significantly not only with the type of method used, whether based on reconstruction error, latent embeddings, or statistical properties also with the nature of the data injections (synthetic anomalies vs. combined anomalies and noise) and the time series length [28]. These discrepancies underscore a critical insight: no single detection approach consistently dominates across all operational scenarios. This heterogeneity in performance reveals the limitations of relying on a one-size-fits-all model in dynamic industrial environments. Instead, manufacturing systems must adopt a multi-model deployment strategy, capable of dynamically selecting the most appropriate detector based on the characteristics of the incoming data stream. For instance, short-duration signals affected by high-frequency noise may benefit from models that emphasize statistical robustness, while longer signals with periodic anomalies might require deep temporal embeddings.

Realizing such a flexible detection pipeline necessitates a scalable, hybrid infrastructure that spans the edge-to-cloud compute continuum. At the network edge, lightweight models must execute with minimal computational overhead to deliver near-instantaneous alerts-crucial for ensuring safety and preventing damage in latency-sensitive settings. These models are typically optimized for real-time inference on constrained hardware, such as industrial gateways or embedded systems. However, edge devices often face limitations due to compute, memory, or power constraints-particularly when dealing with larger models, deep ensembles, or concurrent processing demands. In such cases, the system must intelligently offload feature vectors, latent embeddings, or entire inference tasks to more capable nodes within the fog or cloud infrastructure. These higher-tier platforms can accommodate more expressive, resource-intensive models without compromising throughput or system stability.

To coordinate these decisions effectively, automated orchestration mechanisms are essential. These mechanisms should continuously evaluate contextual factors such as network bandwidth, model memory footprint, real-time inference latency, and the criticality of the alert. Based on these parameters, the system dynamically determines whether to perform inference locally, to partially migrate computation, or to fully offload it upstream. This orchestrated, tiered approach ensures that each anomaly class-regardless of its duration, frequency, or statistical structure handled by the most suitable model, deployed at the most appropriate layer of the infrastructure. As a result, the anomaly detection pipeline remains both adaptable and resilient, capable of meeting the stringent latency, reliability, and scalability demands of modern Industry 5.0 production systems.

5.2. Operator intent translation

As better described in Section 3, a human operator could interface with the system using natural language thanks to the "Human-centric component: LLM-Powered Intent and Policy Processor". This component permits aligning the service placement with the actual need and could be triggered when an anomalous behavior occurs.

Since the ability of this component in understanding the context, the needs, and accurately translating the human intents into machine-readable policies has a notable impact on the overall architecture performance, we conducted a series of experiments to evaluate the accuracy in translating human input. Therefore, we measured the inference time and the accuracy of different LLMs in generating a JSON that has to be syntactically correct and compliant with a predefined schema. In other words, we considered a JSON output valid if it could be deserialized as JSON and had all the required keys (e.g., goal and constraints), and the associated values are the right type. For example, the GPU constraint has to be an integer.

In detail, we considered a JSON output valid if it could be deserialized as JSON, or in other words, has a correct JSON syntax, has all the required keys (e.g., goal and constraints), and the associated values are the right type. For example, the GPU constraint has to be an integer.

For experimentation purposes, we implemented the "*LLM-Powered Intent and Policy Processor*" component using two of the most widely used open-source LLMs, specifically the 8 billion parameter Deepseek-R1 and the 12 billion parameter Gemma3, developed by Google. This choice was made since these two models showed remarkable performance despite their low number of parameters, with respect to others LLMs. Moreover, the low parameter version was chosen since they are more suitable in real industrial environments, where we cannot assume the presence of enough computational resources to run the heavier LLMs. The experiments were conducted on an Ubuntu system featuring 32GB of DDR4 RAM, an Intel Core i7-9750H processor running at 2.60 GHz, and an NVIDIA RTX 3060 GPU with 12GB of VRAM.

We prompted the LLM with a natural language input, simulating an operator request, a requested JSON template, and a list of examples. The requested JSON must contain a list of properties subsequently used to create the policy. These properties include a list of resource constraints and a goal (e.g., minimizing latency).

Table 1
Experimental results across six test sets, each representing a distinct combination of time series duration and injection type. The evaluated anomaly detection strategies include: (i) reconstruction error scoring using an autoencoder (AE) LSTM-based, (ii) latent-space embeddings from AE or Time2Vec models followed by a Local Outlier Factor (LOF) detector, and (iii) statistical properties of the time window combined with LOF. Metrics reported are F_1, $F_{0.5}$, precision, and recall.

TS [s]	Injection	Method	F_1	$F_{0.5}$	Precision	Recall
		AE (recon. error)	1.0	1.0	1.0	1.0
		Latent AE + LOF	0.90	0.90	0.90	0.90
30	A	Time2Vec (recon. error)	0.82	0.86	0.89	0.76
		Latent Time2Vec + LOF	0.86	0.86	0.87	0.85
		Statistic Prop. + LOF	0.66	0.75	0.83	0.56
		AE (recon. error)	0.88	0.82	0.78	1.0
		Latent AE + LOF	0.87	0.86	0.85	0.90
30	$A + N$	Time2Vec (recon. error)	0.81	0.79	0.77	0.85
		Latent Time2Vec + LOF	0.80	0.82	0.83	0.76
		Statistic Prop. + LOF	0.66	0.59	0.56	0.83
		AE (recon. error)	1.0	1.0	1.0	1.0
		Latent AE + LOF	0.83	0.88	0.91	0.77
100	A	Time2Vec (recon. error)	0.95	0.98	1.0	0.91
		Latent Time2Vec + LOF	0.91	0.94	0.97	0.85
		Statistic Prop. + LOF	0.87	0.81	0.77	1.0
		AE (recon. error)	0.86	0.79	0.75	1.0
		Latent AE + LOF	0.72	0.68	0.66	0.81
100	$A + N$	Time2Vec (recon. error)	0.76	0.69	0.65	0.91
		Latent Time2Vec + LOF	0.89	0.92	0.94	0.85
		Statistic Prop. + LOF	0.62	0.51	0.45	0.96
		AE (recon. error)	0.97	0.99	1.00	0.95
		Latent AE + LOF	0.88	0.86	0.86	0.90
180	A	Time2Vec (recon. error)	0.98	0.99	1.0	0.95
		Latent Time2Vec + LOF	0.82	0.82	0.83	0.81
		Statistic Prop. + LOF	0.78	0.70	0.66	0.93
		AE (recon. error)	0.88	0.86	0.82	0.95
		Latent AE + LOF	0.84	0.81	0.79	0.90
180	$A + N$	Time2Vec (recon. error)	0.88	0.84	0.81	0.95
		Latent Time2Vec + LOF	0.83	0.84	0.84	0.81
		Statistic Prop. + LOF	0.80	0.75	0.72	0.90

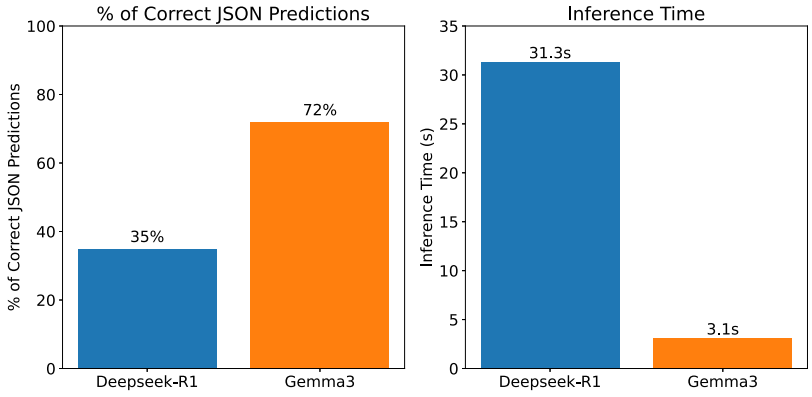


Fig. 6. Diagram showing Gemma3 and Deepseek-R1 accuracy in generating syntactically correct JSON and mean inference time.

The experiments were conducted using 20 different operator requests prompted to the LLM 10 times, to measure the inference robustness. Results are visualized in Fig. 6, which clearly shows how Deepseek-R1 average inference time is more than an order of magnitude larger than the Gemma3 one, because of its "reasoning" process. Despite this, in generating syntactically correct JSONs, Gemma3 performed better, making this LLM more suitable for this kind of activity.

5.3. Service scheduling

We also tested decision-making abilities through the application of a Multi-Criteria Decision-Making (MCDM) method for assessing candidate target nodes for service deployment [44]. Specifically, the objective of this experiment is to evaluate and compare

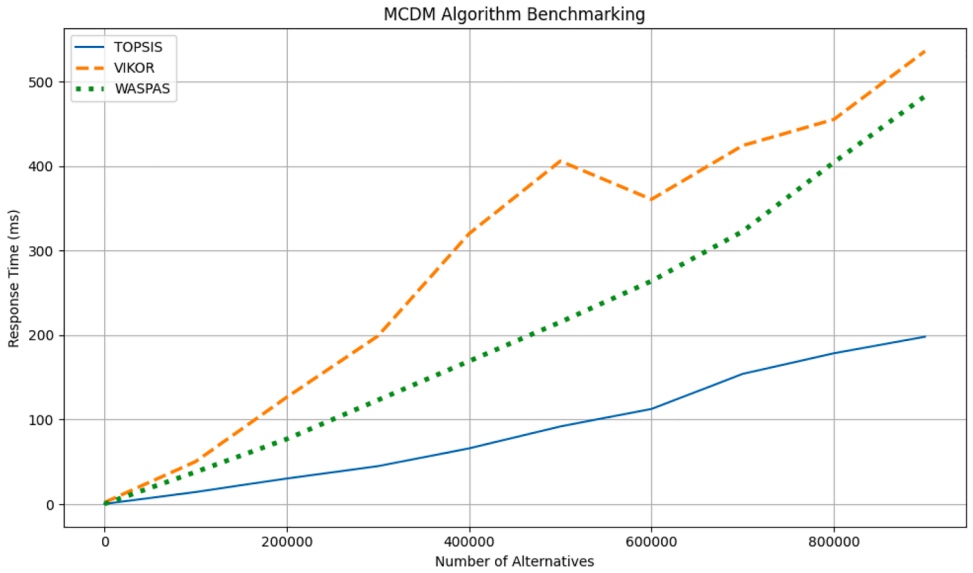


Fig. 7. Computation times of the TOPSIS, VIKOR, and WASPAS when applied on a different number of nodes.

the computational performance of the service scheduler. To reach this goal, we tested four widely-used MCDM algorithms-TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution), VIKOR (VlseKriterijumska Optimizacija I Kompromisno Resenje), and WASPAS (Weighted Aggregated Sum Product ASsessment) - when applied to datasets of increasing size. The primary metric of interest is the response time (in milliseconds) taken by each algorithm to rank synthetic alternatives (nodes) based on multiple criteria. To simulate real-world multi-criteria decision scenarios, we generated synthetic datasets comprising randomly sampled node metrics. Each node was represented as a six-dimensional vector, where each dimension corresponds to a decision criterion. All algorithms were executed using their respective implementations in the pyDecision[1] library. Optional output features such as graphical plots and verbose logs were disabled to minimize overhead and focus purely on core execution time.

Results depicted in Fig. 7 show how all three MCDM algorithms compute extremely fast, even when applied on a large number of nodes. We can also observe some differences between TOPSIS a WASPAS, which perform similarly, and VIKOR. However, all three have performance compatible with an industrial environment.

5.4. Service migration

The following section presents some tests we performed to evaluate the benefits gained from the introduction of components and mechanisms for infrastructure management, service placement, and service migration. Performing these tests, we aim to assess compliance with key industry requirements, such as high availability and security, quantifying some of the key indicators, such as service downtime during service migration, hardware performance, and resource consumption.

5.4.1. Environment set-up

The tests presented below were conducted on a real industrial production line, located at the Bi-Rex company in Bologna, Italy. The infrastructure we configured to test our service management solution includes several nodes on different hardware:

- An edge cabinet hosts edge devices connected to the production machines through a dedicated and isolated network. Below are the hardware specifications of the edge devices:
  - Advantech UNO-148-B - 32 GB Ram: high-performance edge device;
  - NVIDIA Jetson AGX Orin - 12-core Arm CPU, NVIDIA Ampere GPU, 64 GB Ram: high-performance edge device with GPU.
- A virtual machine on a local cloud on the Bi-Rex factory floor (8-core CPU - 32 GB Ram).
- A remote edge device located at the University of Bologna (Lenovo SE360 V2 - Intel Xeon D-2700 - 16 GB Ram).
- A virtual machine on a remote cloud located at the University of Bologna (8-core CPU - 32 GB Ram).

The infrastructure we configured is motivated by the great diversity of industrial-specific devices, located in different places. Although not exhaustive, the analysis aims to be extensive by representing scenarios with different characteristics across various locations (edge, cloud, local, remote). The service being moved within the infrastructure is the anomaly detection application described in Section 4.3. We used both simpler and more complex computational models, which require different data configurations. The goal

[1] https://github.com/Valdecy/pyDecision

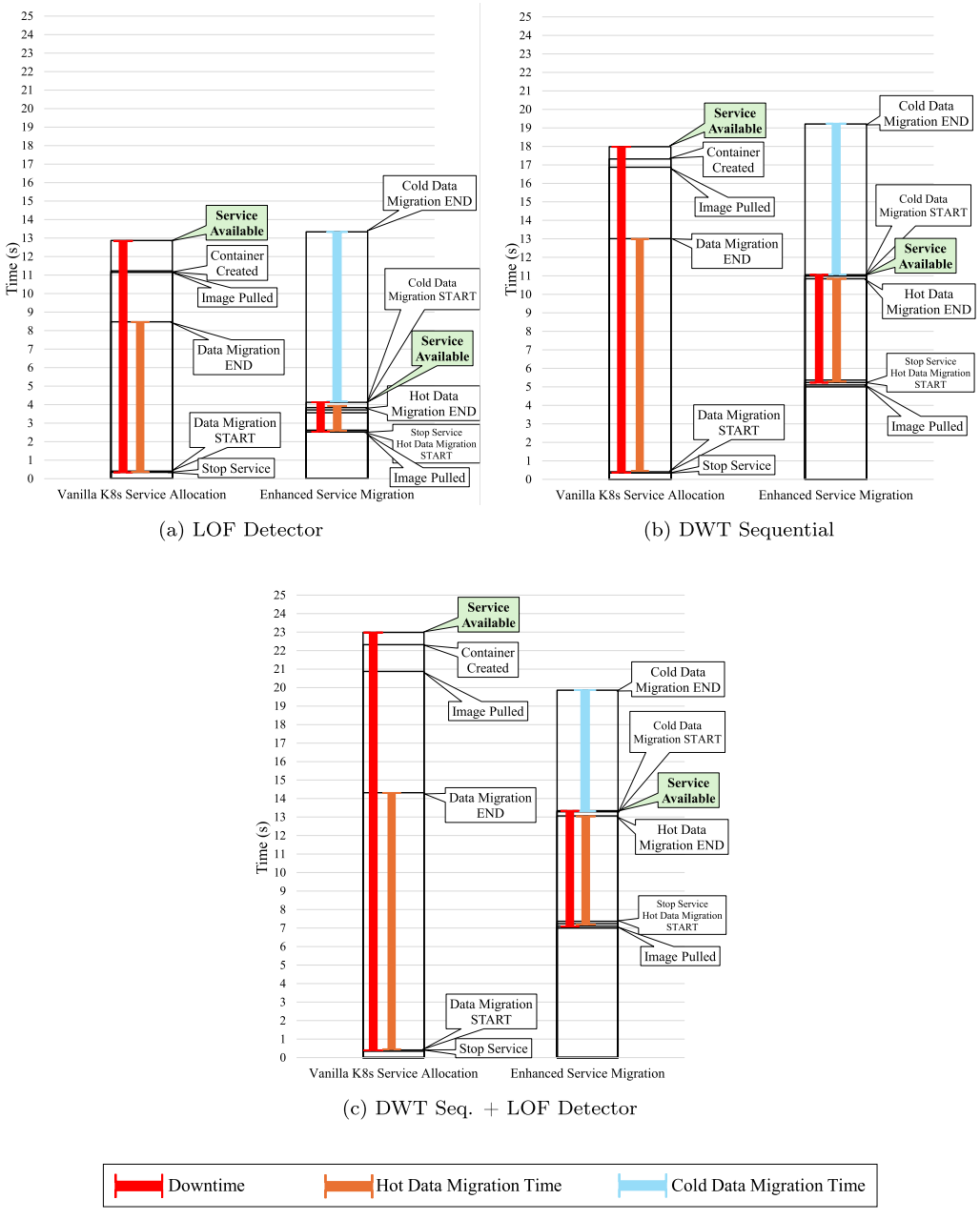


Fig. 8. A detailed analysis of the duration of each migration operation, compared with the default K8s service allocation system.

of our analysis is to assess the response of the proposed service placement and migration system, along with the compliance with both functional and non-functional requirements. Below, the configurations of the anomaly detection algorithms used during the tests are listed. Notably, the elaborated data size of each algorithm is usually defined dynamically and configured during the migration by the Zoom-in and Migration manager component.

- Anomaly Detection using Local Outlier Factor Detector (LOF): analyzes a real-time data stream generated by the machine in the last 30 seconds (approximately 500 KB).
- Anomaly Detection using Sequential DWT model: analyzes a real-time data stream generated by the machine from the last 5 minutes of production (approximately 5 MB).
- Anomaly Detection using both methods, Sequential DWT model + LOF Detector: analyzes a real-time data stream generated by the machine from the last 5 minutes of production (approximately 5 MB).

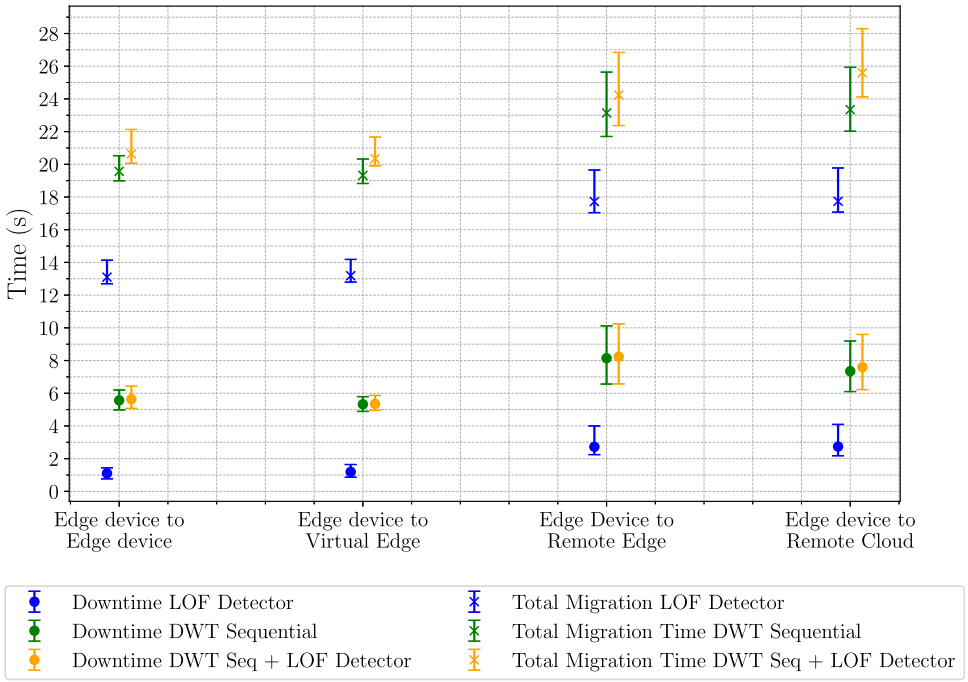


Fig. 9. A comparison between the total migration time and the service downtime during the service migration across the infrastructure.

5.4.2. Migration time and service downtime evaluation

The first set of tests we present aims to evaluate the migration times of the Anomaly Detection service across the various devices in the infrastructure. The main target of our proposal, and relevant in the industrial context, is the reduction of the application's downtime; therefore, it will be the key metric, together with the total migration time.

The first series of tests is conducted by migrating the service between two edge nodes in the Bi-Rex factory floor and is presented in Fig. 8. Each step of the migration is finely analyzed, sampling the duration of each required operation for the service migration, described and represented in Section 3.3 and Fig. 3. The graph in Fig. 8 shows a quantitative comparison with the duration of all the operations performed by the default controller of the vanilla version of K8s. The goal is to highlight the benefits of the proactive migration, the preparatory phase, and the optimized state management presented in our solution, migrating services for anomaly detection. The graph 8a shows the migration times of the anomaly detection service using the *LOF Detector* method. On the left, the migration is carried out by the default K8s controller, and on the right by the proposed migration mechanism. It can be visually observed that the total migration times are similar, while the downtime is significantly lower when using the proposed mechanism. The graph clearly shows that the most time-consuming operations are pulling the service image (from a remote repository) and transferring the state. The mechanism we introduced acts specifically on these two operations, moving them to the background while the service is still available.

The graph Fig. 8(b) shows the migration of an anomaly detection service using a DWT AI model. It is immediately evident how the downtime is affected by the increase in the size of the hot state. The larger model, downloaded from the repository along with the image, also leads to an increase in total migration times. However, it is relevant to note that the dependence on the hot state size is an unavoidable trade-off to maintain infrastructure-level control over migration. In this way, the anomaly detection service starts executing immediately once the hot state is received, unaware of the origin of the resources necessary for execution, which are automatically provided by the infrastructure. Finally, the graph Fig. 8(c) compares the execution of migration operations between the default Kubernetes scheduler and the proposed migration mechanism for a service that combines the two previously analyzed anomaly detection models. Apart from a longer preparation time for retrieving the image, the results are not significantly different from those obtained when running the anomaly detection service using only the DWT Sequential model.

The next graph in Fig. 9 extends the downtime and total migration time analysis, showing relocation tests across the various nodes of the infrastructure. Average values are associated with the variance of the sampled data. As previously observed, the graph highlights a significantly lower downtime compared to the total migration time. Local migrations also exhibit low variance, which slightly increases for heavier models due to the download times of images and models from remote repositories. Similarly, migrations towards remote nodes, whether in the cloud or on edge devices, require longer times due to higher data transfer latency. Network usage increases variance, making the system less predictable. Nevertheless, the limited size of the hot state ensures relatively low downtime with low variance, and thus high availability of the migrated service.

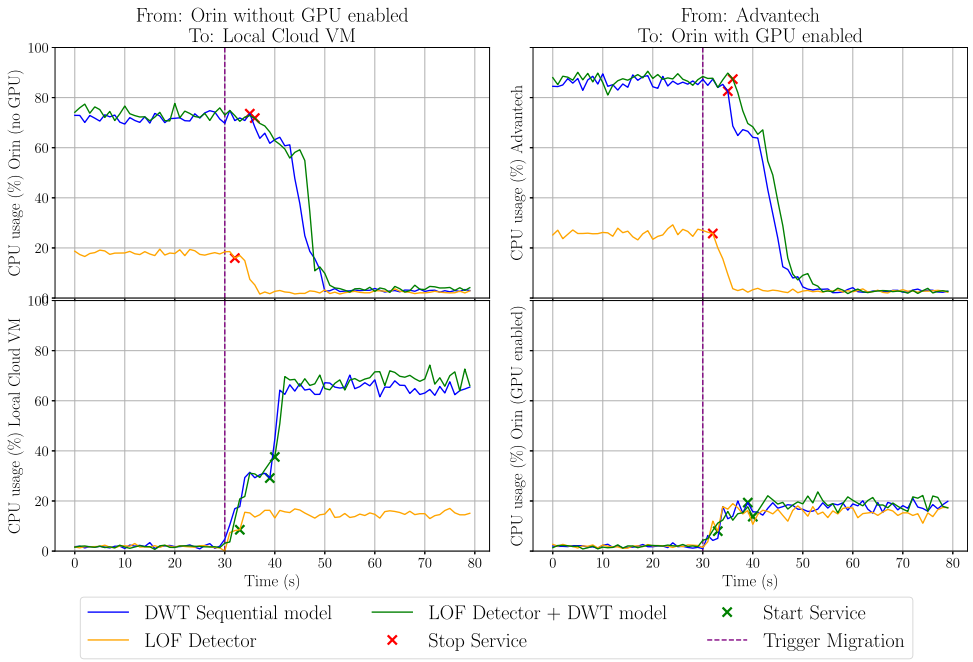


Fig. 10. CPU utilization during execution and service migration of the Anomaly Detection workload on heterogeneous hardware platforms.

5.4.3. Service placement evaluation

In this section, we present the results of several tests designed to stress the infrastructure and trigger migration by running anomaly detection services. The motivation behind this evaluation is to highlight the benefits of efficient migration in an industrial distributed environment. Specifically, in the use case presented in Section 4.3, the anomaly detection service performs collaborative real-time data analysis at the edge, close to the data source, to minimize data transfer and ensure reactive results. However, computational and storage resources at the edge may be limited, precluding the execution of zoomed-in analysis locally. Some edge devices with a GPU are optimal for running machine learning algorithms, but may suffer from limited storage capacity.

The tests presented below result from the migration of anomaly detection services using different models from the edge to the cloud, and from an edge node without GPUs to an edge node equipped with a GPU. In the graph shown in Fig. 10, the CPU usage percentage is reported: the metric is significant for assessing a node's computational availability, stability, and reliability. In the left column, the migration is performed from an edge node with no GPU to a virtual machine in the local cloud. The high-end industrial hardware used offers performance and resources comparable to the virtual machine available in Bi-rex's local cloud. It is important to note, however, that edge resources are limited and non-scalable, unlike the elastic virtual resources available in the cloud. The ability to dynamically migrate services allows for greater system stability and reliability, avoiding node resource saturation and potential failures.

In the right column, we present the migration from one edge node to another equipped with a GPU. The performance in terms of resource consumption for anomaly detection algorithms, especially those using heavy vector-based models, significantly improves thanks to GPU acceleration. As previously emphasized, dynamic service migration, driven by high-level policies, enables the construction of stable and reliable systems. In this case, for instance, heavy model execution, which would have otherwise saturated the edge node, is offloaded to specialized nodes with dedicated hardware.

6. Conclusions and future directions

This paper presented a novel framework for enabling Collective Intelligence within Industry 5.0 environments, introducing a Collective Intelligence-driven stateful service migration mechanism and an innovative Collaborative Intelligence-driven Zoom-In functionality. By extending K8s with intelligent, context-aware service management capabilities and integrating human-in-the-loop decision-making through LLM-driven intent translation, the proposed solution effectively addresses the challenges of service orchestration, resource management, and explainability in dynamic and heterogeneous industrial IoT systems.

Experimental results demonstrate that our framework achieves rapid, reliable service migration with minimized downtime, while enhancing system adaptability and resilience through dynamic AI model upgrading driven by operator expertise. The Zoom-In functionality operationalizes the principles of collaborative intelligence, ensuring that critical decisions are informed by both automated insights and human contextual knowledge.

As future work, we plan to further enhance the framework by integrating predictive analytics and reinforcement learning techniques to proactively anticipate service migration needs. Moreover, we intend to explore the use of lightweight, fine-tuned LLMs for

intent translation. Additional studies will also focus on extending the framework's capabilities toward cross-domain collaborative intelligence scenarios, enabling dynamic coordination among multiple industrial sites and sectors. Lastly, comprehensive validation in larger-scale production environments will be pursued to evaluate scalability, robustness, and long-term operational performance.

CRediT authorship contribution statement

Riccardo Venanzi: Writing - review & editing, Writing - original draft, Visualization, Supervision, Project administration, Methodology, Formal analysis, Data curation, Conceptualization; **Lorenzo Colombi:** Writing - review & editing, Writing - original draft, Visualization, Validation, Software, Data curation; **Davide Tazzioli:** Writing - review & editing, Writing - original draft, Visualization, Validation, Data curation; **Simon Dahdal:** Writing - review & editing, Writing - original draft, Visualization, Validation, Software, Data curation; **Mauro Tortonesi:** Writing - review & editing, Resources, Methodology, Funding acquisition, Formal analysis, Conceptualization; **Luca Foschini:** Writing - review & editing, Resources, Methodology, Funding acquisition, Formal analysis, Conceptualization.

Data availability

Data will be made available on request.

Declaration of competing interest

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

Acknowledgment

This work is partially supported by the European Union - NextGenerationEU - National Recovery and Resilience Plan (Piano Nazionale di Ripresa e Resilienza, PNRR) - Project: "SoBigData.it - Strengthening the Italian RI for Social Mining and Big Data Analytics" - Prot. IR0000013.

References

[1] L. Colombi, A. Gilli, S. Dahdal, I. Boleac, M. Tortonesi, C. Stefanelli, M. Vignoli, A machine learning operations platform for streamlined model serving in industry 5.0, in: NOMS 2024-2024 IEEE Network Operations and Management Symposium, IEEE, 2024, pp. 1–6.
[2] E. Schleiger, C. Mason, C. Naughtin, A. Reeson, C.P. and, Collaborative intelligence: a scoping review of current applications, Appl. Artif. Intell. 38 (1) (2024) 2327890. https://doi.org/10.1080/08839514.2024.2327890
[3] S. Suran, V. Pattanaik, D. Draheim, Frameworks for collective intelligence: a systematic literature review, ACM Comput. Surv. 53 (1) (2020). https://doi.org/10.1145/3368986
[4] A. Bajaj, D.P. Parekh, J.R. Saini, H. Gaikwad, A novel integrated collaborative intelligence framework with innovative BI features for industry 5.0, in: J.C. Bansal, H. Sharma, A. Chakravorty (Eds.), Congress on Smart Computing Technologies, Springer Nature Singapore, Singapore, 2024, pp. 261–277.
[5] I.L.D. Makanda, P. Jiang, M. Yang, H. Shi, Emergence of collective intelligence in industrial cyber-physical-social systems for collaborative task allocation and defect detection, Comput. Ind. 152 (2023) 104006. https://www.sciencedirect.com/science/article/pii/S0166361523001562. https://doi.org/10.1016/j.compind.2023.104006
[6] M. Golovianko, V. Terziyan, V. Branytskyi, D. Malyk, Industry 4.0 vs. Industry 5.0: co-existence, transition, or a hybrid, Procedia Comput. Sci. 217 (2023) 102–113. 4th International Conference on Industry 4.0 and Smart Manufacturing. https://doi.org/10.1016/j.procs.2022.12.206
[7] J. Leng, X. Zhu, Z. Huang, X. Li, P. Zheng, X. Zhou, D. Mourtzis, B. Wang, Q. Qi, H. Shao, J. Wan, X. Chen, L. Wang, Q. Liu, Unlocking the power of industrial artificial intelligence towards industry 5.0: insights, pathways, and challenges, J. Manuf. Syst. 73 (2024) 349–363. https://doi.org/10.1016/j.jmsy.2024.02.010
[8] M. Khosravy, N. Gupta, A. Pasquali, N. Dey, R.G. Crespo, O. Witkowski, Human-Collaborative artificial intelligence along with social values in industry 5.0: a survey of the state-of-the-Art, IEEE Trans. Cognit. Develop. Syst. 16 (1) (2024) 165–176. https://doi.org/10.1109/TCDS.2023.3326192
[9] R.K. de Lima, W.F. Heckler, R. Francisco, J.L. V.B. and, Integrating collaborative learning and advanced technology in industry 5.0: a systematic mapping study and taxonomy, Int. J. Human-Comput. Inter. 41 (1) (2025) 707–722. https://doi.org/10.1080/10447318.2024.2321406
[10] M. Sharma, A. Tomar, A. Hazra, Edge computing for industry 5.0: fundamental, applications, and research challenges, IEEE Internet Things J. 11 (11) (2024) 19070–19093. https://doi.org/10.1109/JIOT.2024.3359297
[11] S. Böhm, G. Wirtz, Cloud-edge orchestration for smart cities: a review of kubernetes-based orchestration architectures, EAI Endorsed Trans. Smart Cities 6 (18) (2022) e2.
[12] P. Kayal, Kubernetes in fog computing: feasibility demonstration, limitations and improvement scope : invited paper, in: 2020 IEEE 6th World Forum on Internet of Things (WF-IoT), 2020, pp. 1–6. https://doi.org/10.1109/WF-IoT48130.2020.9221340
[13] J. Fritzsch, J. Bogner, S. Wagner, A. Zimmermann, Microservices migration in industry: intentions, strategies, and challenges, in: 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2019, pp. 481–490. https://doi.org/10.1109/ICSME.2019.00081
[14] C. Carrión, Kubernetes scheduling: taxonomy, ongoing issues and challenges, ACM Comput. Surv. 55 (7) (2022). https://doi.org/10.1145/3539606
[15] Z. Rejiba, J. Chamanara, Custom scheduling in kubernetes: a survey on common problems and solution approaches, ACM Comput. Surv. 55 (7) (2022). https://doi.org/10.1145/3544788
[16] D. Flores-Martin, J. Berrocal, J. García-Alonso, J.M. Murillo, Towards dynamic and heterogeneous social IoT environments, Computing 105 (6) (2023) 1141–1164.
[17] A.E. Frankó, P. Varga, A survey on machine learning based smart maintenance and quality control solutions, Infocommunications Journal 13 (4) (2021) 28–35.
[18] A. Frankó, G. Hollósi, D. Ficzere, P. Varga, Applied machine learning for IIot and smart production-methods to improve production quality, safety and sustainability, Sensors 22 (23) (2022) 9148.
[19] N. Mäkitalo, A. Ometov, J. Kannisto, S. Andreev, Y. Koucheryavy, T. Mikkonen, Safe, secure executions at the network edge: coordinating cloud, edge, and fog computing, IEEE Softw. 35 (1) (2018) 30–37. https://doi.org/10.1109/MS.2017.4541037
[20] R. Morabito, M. Tatipamula, S. Tarkoma, M. Chiang, Edge AI inference in heterogeneous constrained computing: feasibility and opportunities, in: 2023 IEEE 28th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2023, pp. 225–232. https://doi.org/10.1109/CAMAD59638.2023.10478416

[21] M. Anisetti, C.A. Ardagna, N. Bena, R. Bondaruc, Towards an assurance framework for edge and IoT systems, in: 2021 IEEE International Conference on Edge Computing (EDGE), IEEE, 2021, pp. 41–43.
[22] M. Anisetti, C.A. Ardagna, N. Bena, E. Damiani, Rethinking certification for trustworthy machine-learning-based applications, IEEE Internet Comput. 27 (6) (2023) 22–28. https://doi.org/10.1109/MIC.2023.3322327
[23] S. Dahdal, L. Colombi, M. Brina, A. Gilli, M. Tortonesi, M. Vignoli, C. Stefanelli, An MLOps framework for GAN-based fault detection in bonfiglioli's EVO plant, Infocommun. J. 16 (2) (2024) 2–10. https://doi.org/10.36244/ICJ.2024.2.1
[24] S. Schmidl, P. Wenig, T. Papenbrock, Anomaly detection in time series: a comprehensive evaluation, Proc. VLDB Endow. 15 (2022) 1779–1797.
[25] P. Boniol, Q. Liu, M. Huang, T. Palpanas, J. Paparrizos, Dive into Time-Series Anomaly Detection: A Decade Review, 2024. arxiv.org/abs/2412.20512.
[26] P. Malhotra, L. Vig, G. Shroff, P. Agarwal, Long short term memory networks for anomaly detection in time series, in: ESANN 2015 Proceedings, European Symposium on Artificial Neural Networks, 2015.
[27] J. Miao, H. Tao, H. Xie, J. Sun, J. Cao, Reconstruction-based anomaly detection for multivariate time series using contrastive generative adversarial networks, Inf. Process. Manage. 61 (1) (2024) 103569. https://doi.org/10.1016/j.ipm.2023.103569
[28] L. Colombi, M. Vespa, N. Belletti, M. Brina, S. Dahdal, F. Tabanelli, F. Resca, E. Bellodi, M. Tortonesi, C. Stefanelli, et al., et al., Embedding models for multivariate time series anomaly detection in industry 5.0: l. colombi et al, Data Sci. Eng. (2025) 1–17.
[29] J.L. Herrera, J. Berrocal, H.-Y. Chen, C. Julien, Enabling automated service orchestration in a computing continuum with user-Owned devices, in: 2024 IEEE International Conference on Software Services Engineering (SSE), 2024, pp. 86–96. https://doi.org/10.1109/SSE62657.2024.00024
[30] L. Colombi, I. Boleac, M. Brina, S. Dahdal, M. Tortonesi, M. Vignoli, C. Stefanelli, Multi-Cluster MLOps platform for industry 5.0, in: 2025 IEEE Symposium on Computers and Communications (ISCC), 2025.
[31] P. Bellavista, S. Dahdal, L. Foschini, D. Tazzioli, M. Tortonesi, R. Venanzi, Kubernetes enhanced stateful service migration for ML-Driven applications in industry 4.0 scenarios, in: 2024 IEEE Annual Congress on Artificial Intelligence of Things (AIoT), IEEE, 2024, pp. 25–31.
[32] A. Calagna, Y. Yu, P. Giaccone, C.F. Chiasserini, Design, modeling, and implementation of robust migration of stateful edge microservices, IEEE Trans. Netw. Serv. Manage. 21 (2) (2024) 1877–1893. https://doi.org/10.1109/TNSM.2023.3331750
[33] H. Koziolek, A. Burger, A. Puthan Peedikayil, Fast state transfer for updates and live migration of industrial controller runtimes in container orchestration systems, J. Syst. Softw. 211 (2024) 112004. https://doi.org/10.1016/j.jss.2024.112004
[34] A.B.A. Alaasam, G. Radchenko, A. Tchernykh, J.L. González Compeán, Analytic study of containerizing stateful stream processing as microservice to support digital twins in fog computing, Programm. Comput. Softw. 46 (2020) 511–525.
[35] C. Benzaid, T. Taleb, AI-driven zero touch network and service management in 5G and beyond: challenges and research directions, IEEE Netw. 34 (2) (2020) 186–194.
[36] M. Liyanage, Q.-V. Pham, K. Dev, S. Bhattacharya, P.K.R. Maddikunta, T.R. Gadekallu, G. Yenduri, A survey on zero touch network and service management (ZSM) for 5G and beyond networks, J. Netw. Comput. Appl. 203 (2022) 103362. https://www.sciencedirect.com/science/article/pii/S1084804522000297. https://doi.org/10.1016/j.jnca.2022.103362
[37] ETSI, Zero-touch network and Service Management (ZSM); Intent-driven autonomous networks;Generic aspects, Technical Report ETSI GR ZSM 011 V1.1.1, European Telecommunications Standards Institute (ETSI), 2024. https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/018/01.01.01_60/gs_ZSM018v010101p.pdf.
[38] A. Clemm, L. Ciavaglia, L.Z. Granville, J. Tantsura, Rfc 9315: intentbased networking - concepts and definitions, 2022.
[39] K. Dzeparoska, J. Lin, A. Tizghadam, A. Leon-Garcia, LLM-based policy generation for intent-based management of applications, in: 2023 19th International Conference on Network and Service Management (CNSM), IEEE, 2023, p. 1-7. https://doi.org/10.23919/cnsm59352.2023.10327837
[40] K. Dzeparoska, N. Beigi-Mohammadi, A. Tizghadam, A. Leon-Garcia, Towards a self-driving management system for the automated realization of intents, IEEE Access 9 (2021) 159882–159907. https://doi.org/10.1109/ACCESS.2021.3129990
[41] J. Massa, S. Forti, F. Paganelli, P. Dazzi, A. Brogi, Declarative provisioning of virtual network function chains in intent-based networks, in: 2023 IEEE 9th International Conference on Network Softwarization (NetSoft), 2023, pp. 522–527. https://doi.org/10.1109/NetSoft57336.2023.10175449
[42] D. Tazzioli, R. Venanzi, L. Foschini, Stateful service migration support for kubernetes-based orchestration in industry 4.0, in: 2024 IEEE Symposium on Computers and Communications (ISCC), IEEE, 2024, pp. 1–6.
[43] R. Venanzi, S. Dahdal, M. Solimando, L. Campioni, A. Cavalucci, M. Govoni, M. Tortonesi, L. Foschini, L. Attana, M. Tellarini, C. Stefanelli, Enabling adaptive analytics at the edge with the Bi-Rex big data platform, Comput. Ind. 147 (2023) 103876. https://doi.org/10.1016/j.compind.2023.103876
[44] I. Dimolitsas, D. Dechouniotis, S. Papavassiliou, P. Papadimitriou, V. Theodorou, Edge cloud selection: the essential step for network service marketplaces, IEEE Commun. Mag. 59 (10) (2021) 28–33. https://doi.org/10.1109/MCOM.211.2001056