A Scalable Fog Computing Solution for Industrial Predictive Maintenance and Customization

electronics

Article

A Scalable Fog Computing Solution for Industrial Predictive Maintenance and Customization

Pietro D'Agostino [1,*], Massimo Violante [1] and Gianpaolo Macario [2]

[1] Department of Control and Computer Engineering, Politecnico di Torino, Corso Castelfidardo, 34/d, 10138 Torino, Italy; massimo.violante@polito.it
[2] AROL Closure Systems, Viale Italia, 193, 14053 Canelli, Italy; gianpaolo.macario@arol.com
* Correspondence: pietro.dagostino@polito.it

Abstract: This study presents a predictive maintenance system designed for industrial Internet of Things (IoT) environments, focusing on resource efficiency and adaptability. The system utilizes Nicla Sense ME sensors, a Raspberry Pi-based concentrator for real-time monitoring, and a Long Short-Term Memory (LSTM) machine-learning model for predictive analysis. Notably, the LSTM algorithm is an example of how the system's sandbox environment can be used, allowing external users to easily integrate custom models without altering the core platform. In the laboratory, the system achieved a Root Mean Squared Error (RMSE) of 0.0156, with high accuracy across all sensors, detecting intentional anomalies with a 99.81% accuracy rate. In the real-world phase, the system maintained robust performance, with sensors recording a maximum Mean Absolute Error (MAE) of 0.1821, an R-squared value of 0.8898, and a Mean Absolute Percentage Error (MAPE) of 0.72%, demonstrating precision even in the presence of environmental interferences. Additionally, the architecture supports scalability, accommodating up to 64 sensor nodes without compromising performance. The sandbox environment enhances the platform's versatility, enabling customization for diverse industrial applications. The results highlight the significant benefits of predictive maintenance in industrial contexts, including reduced downtime, optimized resource use, and improved operational efficiency. These findings underscore the potential of integrating Artificial Intelligence (AI) driven predictive maintenance into constrained environments, offering a reliable solution for dynamic, real-time industrial operations.

Keywords: fog computing; industrial IoT; predictive maintenance; LSTM; platform integration; industrial applications

Academic Editor: Andrea Bonci

Received: 9 December 2024
Revised: 17 December 2024
Accepted: 22 December 2024
Published: 25 December 2024

Citation: D'Agostino, P.; Violante, M.; Macario, G. A Scalable Fog Computing Solution for Industrial Predictive Maintenance and Customization. *Electronics* **2025**, *14*, 24. https://doi.org/10.3390/electronics14010024

1. Introduction

The rapid proliferation of devices and data-intensive applications in the Internet of Things (IoT) presents significant challenges for industries in managing and processing large amounts of data [1]. As the volume of data generated at the edge increases, traditional cloud computing solutions face limitations, particularly latency. Cloud computing typically involves sending data to centralized servers far from the edge, which introduces delays due to the distance between devices and cloud data centers [2–4].

Edge computing aims to address this by processing data directly at or near the source of data generation, either on the devices themselves or at local edge nodes, minimizing latency and reducing the load on centralized cloud systems. However, while edge computing improves responsiveness by processing data locally, it can be limited in its ability to manage large-scale data and complex tasks due to constraints in processing power, storage, and connectivity [5].

Fog computing builds on edge computing by creating an intermediate layer between edge devices and the cloud [6], like represented in Figure 1. Fog nodes, placed closer to the data sources, process and store data locally while maintaining the flexibility and scalability of the cloud [7]. This hybrid architecture improves the performance of real-time data processing and reduces latency more effectively than cloud-based solutions alone [8]. Moreover, by managing data processing closer to the edge and at multiple points in between, fog computing reduces the need to transmit large volumes of data to the cloud, thus improving privacy and minimizing network congestion [9]. The disadvantages of fog computing were carefully evaluated compared to edge and cloud computing [10]. It was concluded that fog computing offered the most suitable solution for this specific application. This choice was driven by the need to manage a high number of sensors, which would generate an excessive data flow for cloud computing, and edge computing was not feasible due to the limited space available within the capping machine.

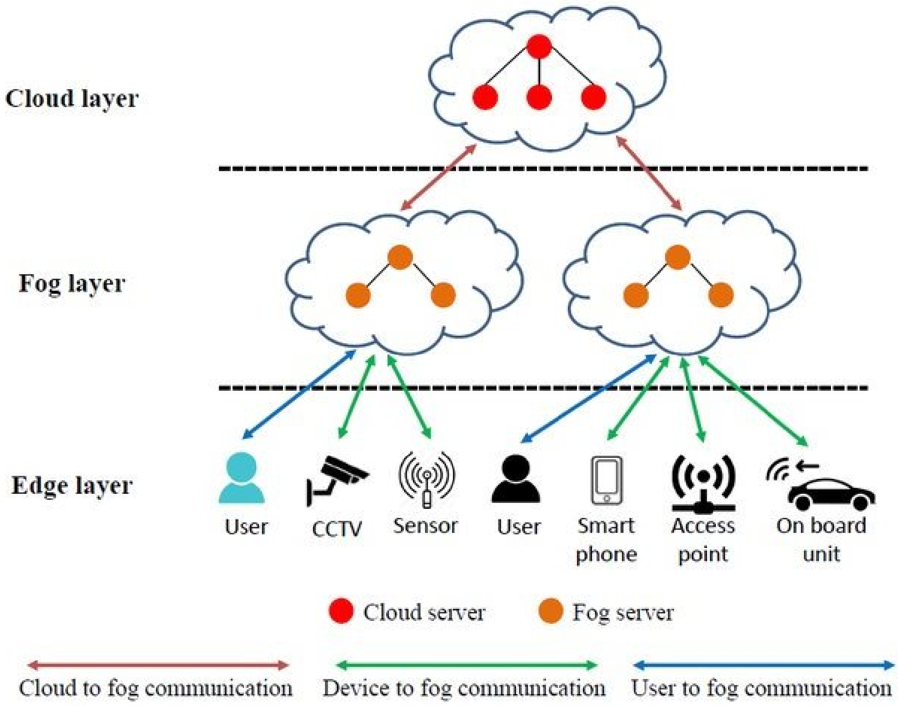

Figure 1. Fog computing architecture [11].

Although fog computing is particularly beneficial for industrial environments, where real-time data processing and low latency are crucial, its applications extend to nonindustrial sectors such as smart cities [12], transportation [13], and environmental or health monitoring [14]. These systems benefit from reduced reliance on centralized cloud services, which improves efficiency and responsiveness [15]. However, a key limitation of current fog solutions is that they often lack customization to meet the specific needs of industrial applications, where high levels of customization are required to integrate a wide range of devices, sensors, and predictive models [16]. The need for more flexibility in existing systems is one of the central issues that this paper aims to address.

The flexibility of these systems, while offering benefits such as improved resource optimization and adaptability [17], also increases vulnerability to cyber threats [18]. Distributed denial of service (DDoS) attacks [19] and data breaches [20] could be some examples. DDoS attacks can overwhelm network resources, causing service disruptions or delays in real-time data collection and anomaly detection [21]. In contrast, data breaches can lead to unauthorized access to sensitive information, compromising operational integrity and regulatory compliance [22].

IIoT systems must be designed with adaptable security protocols to address these evolving risks. The customization capabilities of these systems provide an opportunity to integrate tailored defense mechanisms that help mitigate specific threats. For example, encryption techniques (such as AES-256 [23]) and multi-factor authentication (MFA) [24] can be employed to secure data both at rest and in transit, while traffic monitoring and anomaly detection algorithms can identify and mitigate DDoS attempts in real-time [25].

By aligning security measures with operational needs, IIoT systems can provide reliable predictive maintenance and remain resilient to potential interruptions. This ability to integrate robust security at every system layer—especially within sandbox environments that allow for external customization—ensures the platform can withstand known and emerging threats, ultimately protecting critical industrial operations.

To address these challenges, this paper introduces the concentrator platform, which directly addresses key limitations of fog computing in the context of predictive maintenance. One of the primary issues in traditional fog computing systems is their lack of sufficient customization options, especially when handling complex industrial environments with a diverse array of devices and sensors. In response, the concentrator provides a highly flexible solution through Docker-based containerization, allowing easy integration of specific sensors and algorithms tailored to operational needs [26]. Docker allows isolated containers to operate independently on the same kernel, enabling efficient management of resources without interference unless specifically allowed [27]. This modular approach enhances the system's adaptability and performance in environments where customization is critical for success. Moreover, the concentrator also addresses the issue of managing large-scale data generated by numerous devices. By utilizing fog nodes for localized data processing and leveraging scalable cloud resources, the platform effectively balances the need for real-time data analysis with the demands of large data volumes [28]. Unlike edge computing, which can struggle with limited processing power and storage capacity, the concentrator's distributed architecture enables efficient management of complex tasks without overburdening the cloud infrastructure.

AROL Closure Systems (Canelli, Italy), a manufacturer of capping machines, demonstrated the system using long short-term memory (LSTM) models for predictive maintenance. By predicting temperature anomalies and scheduling maintenance before equipment failures occur, the system helps optimize machine uptime [29]. LSTM models excel in forecasting time-series data by capturing long-term dependencies, offering advantages over traditional predictive models [30–32].

The main contributions of this paper are as follows:

- The development and deployment of concentrators in a real-world industrial setting.
- Customization capabilities for integrating specific sensors and algorithms.
- The application of LSTM models to enhance predictive maintenance and reduce downtime.

This paper is structured as follows: Section 2 reviews related work on fog computing and predictive maintenance. Section 3 outlines the design and implementation of the concentrator system, with an emphasis on its customization and machine learning features. Section 4 discusses the real-world deployment of the system and its performance in a production environment. Section 5 compares existing infrastructure with the results obtained, and finally, Section 6 presents some conclusions and future works.

2. State of Art

This work primarily revolves around two key aspects: device customization and predictive maintenance functionalities, all developed through a fog computing system.

Researchers often focus on device customization by laying the groundwork for diverse application scenarios. For instance, Isaac Lera et al. [33] developed YAFS, a dynamic fog computing system enabling edge components to select the nearest and most secure link with various nodes, ensuring stable connections. Despite its utility for highly dynamic systems, YAFS does not add anything about the user's application, even though it can be useful for highly dynamic systems.

Cecil Wöbker et al. [34] presented Fogernetes, a framework aimed at streamlining the deployment and management of fog computing applications. Fogernetes offers tailored features for fog environments, including edge node discovery, resource-aware scheduling, and dynamic adaptation to network conditions. While the authors successfully demonstrate the system's effectiveness, its adaptability is contingent upon the network; customers have limited scope for additional customization.

D. D'Alessandro et al. described a similar project in [35], exploring mass customization in IoT wireless sensor systems. This study focuses on the need for flexible IoT solutions that adapt to different application requirements and environments. The authors presented a modular approach to designing and developing IoT sensor systems, which allows for flexibility and scalability during deployment. Integrating modular components like sensors, communication modules, and processing units allows the system to be customized to meet specific user needs without requiring major redesigns or reconfigurations.

Unlike traditional methods that rely on hubs and separate radio modules, the concentrator emphasizes embedding custom firmware directly into embedded systems used in production lines. This streamlined approach improves efficiency and reduces the need for additional hardware components unless required by the client. Additionally, by using Docker and containerization, as previously mentioned, it is possible to create adaptable environments tailored to each customer's needs. This method provides exceptional flexibility and customization options, making integrating into various environments and applications accessible. This solution also adapts to microservices and IoT scenarios on embedded systems, such as a Raspberry Pi; Ref. [36] presents an example of its usage, making the solution easily deployable in production lines.

Another important aspect to consider is federated learning (FL), which has emerged as a transformative approach in decentralized machine learning, particularly for applications requiring data privacy, scalability, and efficiency. Recent studies demonstrate the potential of FL in various domains, such as energy management and intelligent edge computing. One study introduces a federated learning gradient boosting (FedXGB) framework integrated with edge computing for solar generation forecasting in residential power systems [37]. This solution, leveraging FogFlow for orchestrating IoT ecosystems, addresses critical issues like data privacy, operational efficiency, and network optimization, showing remarkable results, especially in mid and long-term solar forecasting. By decentralizing the forecasting process, FL ensures that energy data remains on local devices, safeguarding privacy while optimizing energy efficiency in residential environments.

Another study optimizes intelligent edge computing resource scheduling through FL, tackling challenges such as device heterogeneity, non-IID data distribution, and communication overhead [38]. The proposed framework includes an adaptive client selection mechanism that considers computational power, energy status, data quality, multi-task learning, and local batch normalization layers to handle non-IID data. Communication-efficient updates are incorporated to reduce bandwidth consumption and implement a robust privacy policy to enhance data protection. The framework demonstrates a 15% improvement in model accuracy and a 40% reduction in communication overhead compared to conventional FL methods, showcasing its effectiveness in real-world applications like intelligent city traffic prediction and healthcare IoT.

In contrast to federated learning, the system in this study adopts a different approach: an external source creates and trains the model, while the Raspberry Pi is used solely for local inference based on the pre-trained model. The model's predictions are modified depending on the data from the sensor nodes, which provides some level of customization but without the Raspberry Pi contributing to the training or updates of the model. While federated learning involves decentralized, collaborative model training across multiple devices, the Raspberry Pi only performs inference, limiting its ability to learn or improve over time.

Shifting the focus to predictive maintenance, Ref. [39] illustrated the critical significance of these methods for industries aiming to minimize downtime, cut costs, prolong equipment lifespan, enhance operational efficiency, and support safety measures. The authors employ charts to assess the return on investment (ROI) based on commonly used variables across various industries. However, they highlight a significant challenge: the indispensable requirement for high-quality data to ensure dependable predictions. They emphasize the crucial role of meticulous data preparation and cleaning in guaranteeing the reliability of predictive maintenance systems. These results underscore the importance of data quality in deriving actionable insights and maximizing the effectiveness of predictive maintenance strategies.

Both [40,41] explored various machine learning methods and demonstrated their effectiveness in performing machine predictive maintenance. However, their focus primarily centered on classification techniques, including support vector machines, logistic regression, k-nearest neighbors, random forest, Gaussian Naïve Bayes, decision trees, and neural networks. Based on the study, researchers have considered the LSTM the optimal solution for predicting values from temporal sequences.

Another example of the usage of AI alongside an embedded system can be seen in Ref. [42], where F. Laganà presented an integrated system for surface electromyography (sEMG) signal acquisition, processing, and analysis developed using artificial intelligence (AI) techniques. The study addressed the challenge of analyzing noisy and complex sEMG signals by leveraging the computational capabilities of the Raspberry Pi. The author demonstrates how this affordable platform can process real-time data while running machine-learning algorithms for classifying muscle activity. The system's modular design allows easy integration with various applications, such as myoelectric prostheses and robotic control systems, while maintaining flexibility and scalability.

As demonstrated in the following studies, researchers have developed machine learning and predictive maintenance on a fog computing system.

As discussed in [43], the authors presented a comprehensive survey on deploying machine learning (ML) on edge devices and cloud networks. The paper explored various IoT devices, including Raspberry Pi, NVIDIA Jetson, and Arduino Nano 33 BLE Sense, which integrate edge intelligence for object detection and gesture recognition tasks. The study also analyzes 1000 recent publications on "ML in IoT" to identify emerging topics and application domains, highlighting challenges such as security, privacy, and energy limitations in edge device implementation. The survey provides an in-depth look at traditional ML and deep learning methods in IoT contexts, showcasing their potential in diverse application domains, including healthcare, agriculture, and smart cities.

In [44], the authors explored the use of edge computing (EC) for anomaly detection in Internet of Things (IoT) environments, focusing on the integration of machine learning (ML) and deep learning (DL) models with microcontroller units (MCUs). The paper provided a systematic literature mapping of 18 studies published between 2021 and 2023, addressing the benefits and challenges of anomaly detection using TinyML on MCUs. The survey investigated different ML/DL algorithms, validation metrics, data estimation techniques, and hardware/software configurations while proposing a taxonomy of algorithms used in TinyML applications. The findings demonstrate how researchers can apply ML effectively in EC environments to ensure accurate data and improve the reliability of IoT systems despite the unique challenges posed by IoT devices' resource constraints.

Ref. [45] presented a detailed analysis of using reinforcement learning (RL) and deep reinforcement learning (DRL) for offloading tasks in fog computing environments. The paper highlighted how RL and DRL can be applied to manage the offloading of computationally intensive tasks from resource-constrained IoT devices to fog or cloud layers to reduce latency and improve system performance. The authors discussed various offloading mechanisms, categorizing them into value-based, policy-based, and hybrid-based algorithms, and comparing them based on offloading modes, performance metrics, and evaluation tools. The study also identified open research areas and future directions, providing insights into how DRL can help overcome the challenges of large-scale networks and complex state-action spaces.

An example of actual applications can be seen in Ref. [46], where the authors proposed a novel fog data prediction and recovery (FDPR) algorithm. This algorithm leverages deep learning techniques, specifically a deep concatenated multilayer perceptron (DC-MLP), to predict and recover missing sensor data in a fog computing layer of an IoT network. By efficiently addressing data prediction and recovery, the proposed algorithm achieves high accuracy (99.89%) and significantly improves IoT device lifetime (121%) while maintaining low overhead. The evaluation, conducted through simulations and experimental work with nine-edge devices, demonstrates its effectiveness in real-world scenarios.

Another example can be found in Ref. [47], where a fog-enabled air quality monitoring and prediction (FAQMP) system was proposed to address challenges in air quality monitoring and forecasting in smart cities. The system integrates IoT, fog computing (FC), low-power wide-area networks (LPWANs), and deep learning (DL) to improve the accuracy and efficiency of air quality predictions. A lightweight DL-based Seq2Seq gated recurrent unit (GRU) attention model, optimized through post-training quantization, was employed for multi-step forecasting. The model demonstrated superior performance with reduced computational load, making it suitable for deployment on resource-constrained fog nodes. The FAQMP system provides real-time air quality updates, forecasts, and alerts through the EnviroWeb application, assisting governments in maintaining air quality standards and fostering a healthier environment.

A novel decentralized air quality prediction and event detection system, DeepFogAQ, was proposed in [48] to tackle the issue of air pollution in future cities. The system predicts pollutant concentrations and detects environmental events by integrating deep learning models, fog computing, and complex event processing. The architecture utilizes various models, such as Transformers, CNN-LSTM, and GRU, to ensure accurate predictions. Experimental results highlight the superior performance of the Transformer model in air quality forecasting. DeepFogAQ's decentralized, scalable, and fault-tolerant structure offers a promising solution for managing air pollution and supporting decision-making processes.

While these studies effectively utilize LSTM networks and machine learning in similar contexts, the system proposed in this work prioritizes adaptability. The platform facilitates easy integration of custom machine learning models through its sandbox environment, offering enhanced versatility for diverse industrial applications.

To optimize LSTM models for these environments, researchers can employ several techniques, as follows:

- **Model pruning and quantization:** Reducing the size of the LSTM model through pruning unnecessary parameters and applying quantization techniques helps decrease the memory footprint and computational load, making the model more suitable for embedded systems [49].
- **Knowledge distillation**: This involves training a smaller, less complex model (student model) to mimic the behavior of a larger, more complex LSTM model (teacher model). The smaller model can then be deployed on resource-constrained devices without significant loss of accuracy [50].
- **Hardware acceleration**: Utilizing specialized hardware, such as tensor processing units (TPUs) or field-programmable gate arrays (FPGAs), can significantly accelerate the computation of LSTM models on embedded devices, enabling real-time processing [51].
- **Edge AI frameworks**: Using edge AI frameworks like TensorFlow Lite or ONNX Runtime helps convert and optimize LSTM models for execution on embedded systems, ensuring efficient performance while maintaining high accuracy [52].

By implementing these optimization strategies, industries can effectively deploy LSTM-based predictive maintenance solutions on embedded systems, ensuring robust and reliable performance even in resource-constrained environments [53]. In this case, a combination of model pruning, quantization, and hyperparameter tuning was applied to strike an optimal balance between computational efficiency and prediction accuracy. This strategy enables the deployment of highly efficient models without compromising the reliability of predictive maintenance, making them ideal for resource-constrained environments. Additionally, these optimizations help manage memory usage, ensuring the model retains only the essential features for accurate predictions while avoiding unnecessary information overload.

Compared to these articles, the LSTM model of the concentrator serves a dual purpose within the system. First, it is utilized for predictive maintenance, ensuring the machine operates correctly and alerting the user to potential issues if specific parameters deviate from expected values. Second, the model monitors the sensors' performance. Since the system is homogeneous, if predictions from one sensor diverge significantly from those of its neighboring sensors, it may indicate a problem with the measurement system itself. Therefore, the LSTM model aids in verifying both the machine's proper operation and the accuracy of the measurement system.

Numerous studies have attested to the superior performance of LSTM models compared to their counterparts, showcasing their ability to achieve remarkable accuracy and generalization on diverse datasets [54]. LSTM's adaptability to varying data complexities and ability to learn intricate patterns make it a favored choice for addressing real-world industry challenges. Its versatility and proven efficacy solidify LSTM's position as a key technology in deep learning [55].

While the LSTM model excels in predictive maintenance due to its ability to capture long-term dependencies, deploying such models on embedded systems poses unique challenges. Embedded systems, often characterized by limited computational power, memory, and energy resources, require efficient deployment of LSTM models to ensure real-time performance without exhausting system resources [56].

3. Methodology and Architecture

3.1. The Native System

This section will describe the entire system, showing the architecture of the concentrator, as seen in Figure 2, and its functionalities, as seen in Figure 3.

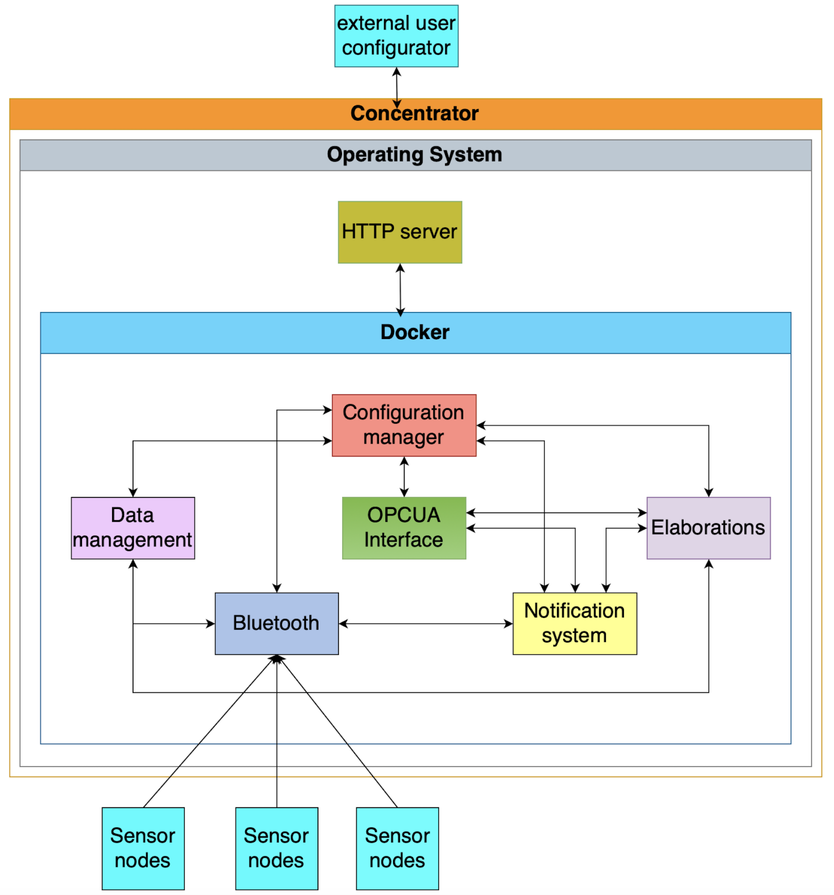

Figure 2. The native system of the concentrator, without the sandbox.

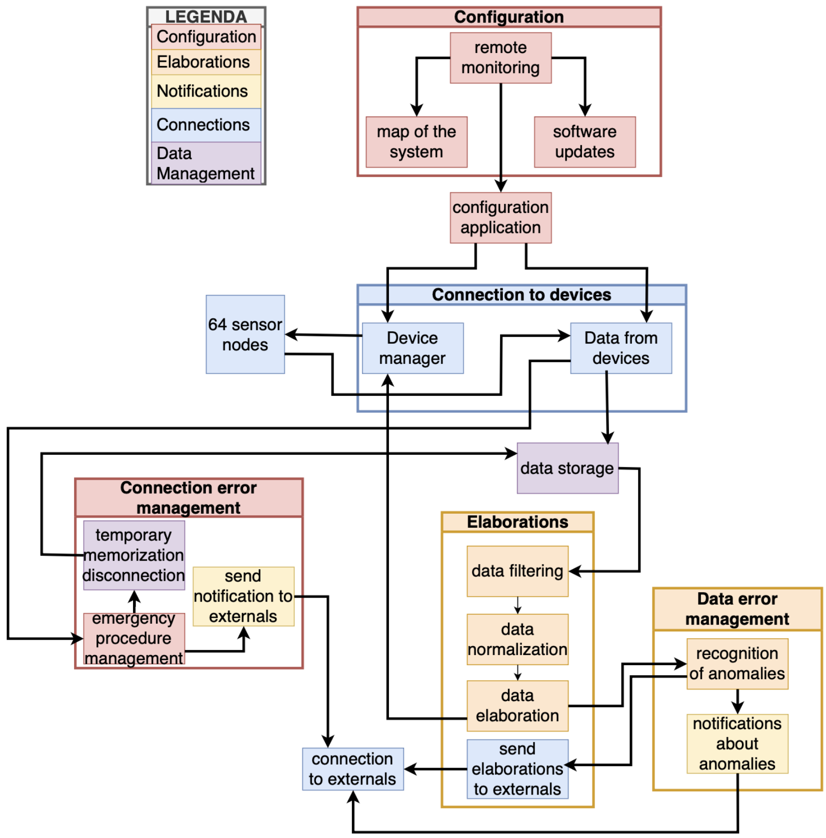

Figure 3. Functions of the native system of the concentrator, without the sandbox.

The concentrator system is a comprehensive framework for real-time data management and analysis within industrial environments. The essential functionalities or use cases (UCs) include the following:

- **UC.1**: Wireless data retrieval from sensors and devices.
- **UC.2**: Versatile handling of process data from AROL (Canelli, IT) and external systems.
- **UC.3**: Rigorous technical data processing for real-time monitoring and optimization.
- **UC.4**: Prompt anomaly detection with effective user notification.
- **UC.5**: Multi-device connectivity for holistic monitoring.

These functionalities collectively empower the concentrator to serve as a central data aggregation, processing, and dissemination hub, facilitating informed decision-making, predictive maintenance, and operational optimization. Moreover, with provisions for seamless integration of custom sensors and adapters, the concentrator ensures interoperability across diverse industrial ecosystems.

- *Protocol requirements*
  The concentrator system employs robust communication protocols tailored for seamless interaction with IoT and operational technology (OT) systems in industrial settings. For IoT device communication, it utilizes wireless connectivity, supporting protocols like Bluetooth Low Energy (BLE) version 4.0 and Wi-Fi. The Wi-Fi capability enables transmission using various protocols such as message queuing telemetry transport (MQTT), hypertext transfer protocol (HTTP), transmission control protocol/internet protocol (TCP/IP), and constrained application protocol (CoAP) [57]. Notably, the platform handles multiple connections from IoT devices, handling connection establishment, maintenance, and termination while automatically reconnecting during temporary disruptions. While communication with OT systems is pending development, the concentrator will interface with the Modbus transmission control protocol (Modbus/TCP) and OPC unified architecture (OPCUA) protocol, facilitating data exchange with industrial automation devices such as programmable logic controllers (PLCs) and supervisory control and data acquisition (SCADA) systems. Additionally, it features synchronization inputs for the temporal alignment of devices or actions, ensuring precise coordination within industrial processes. Moreover, potential integration of Ethernet-based protocols may be necessary to accommodate client-specific devices, highlighting the concentrator's adaptability to diverse industrial environments.
- *Functional requirements*
  The concentrator system has various functionalities essential to operate within industrial contexts. Firstly, the system addresses the platform power management requirement, including startup, shutdown, and configuration procedures. The concentrator then establishes robust connections with various external systems regarding system communication. Notably, the concentrator can communicate simultaneously with a maximum of 64 sensor node devices (usually mounted on a capping machine) while maintaining connectivity to additional devices as required. Moreover, to ensure future scalability and adaptability, compatibility with diverse IoT devices can be possible as long as the hardware limitations permit it.
- *Data Processing*
  The concentrator system plays a crucial role in enhancing the quality and utility of data received from IoT devices. Initially, it performs preprocessing on data from IoT devices to ensure compatibility and efficiency before transmitting it to external systems. Depending on the user's configuration, it also filters data based on specific criteria, potentially removing invalid, out-of-range, or non-compliant data to maintain data integrity and relevance. Additionally, the system aggregates data from various IoT devices to offer a comprehensive overview of collected information, aiding informed

decision-making. It can also apply calibrations or normalizations to ensure data aligns with external system standards, improving interoperability.

- *Error Management*
  The concentrator is responsible for robust error detection and mitigation to maintain operational reliability and data integrity. It identifies errors and anomalies like connection failures, transmission errors, out-of-range data, or invalid values. Once detected, the system communicates these errors via messages, system logs, or other notifications for timely intervention and resolution. Additionally, it can address critical errors through emergency procedures such as system shutdown or activating safety modes to prevent potential hazards. Moreover, the concentrator supports temporary data storage during system disconnection, ensuring uninterrupted data transmission upon reconnection.
- *Cybersecurity and Updates and Maintenance*
  The primary responsibility of the concentrator is to authenticate IoT devices, ensuring that only authorized devices can communicate with the network. This authentication process is pivotal in safeguarding sensitive data from unauthorized access, thereby preserving the confidentiality and integrity of the system. Moreover, the platform manages data access dynamically, preventing potential data breaches and ensuring that only permitted users or devices can interact with sensitive information.
  In addition to authentication and access control, the system supports remote administration and monitoring, enabling the continuous execution of system updates and maintenance. This capability ensures the platform remains up-to-date with the latest security patches, protecting it from emerging vulnerabilities. The system can proactively identify issues and optimize performance through remote monitoring, maintaining robust security and functionality. This approach to remote control also facilitates timely intervention for system enhancements and problem resolution, ensuring and introducing continuous security measures to address evolving cyber threats.

3.2. The Host System

The host system is designed as a sandbox environment, enabling users to customize its functionality by integrating external machine learning models or algorithms. This adaptability enhances the system's usability across various industrial applications. As an example, this paper demonstrates the integration of an LSTM model for predictive maintenance. The model processes sensor temperature data to forecast future values in real-time, enabling proactive operational decisions.

Time series forecasting models for resource-constrained embedded systems have been extensively studied, each offering distinct trade-offs. Traditional methods such as ARIMA [58] and exponential smoothing (ETS) [59] are computationally efficient and suitable for more straightforward, stationary datasets. However, their inherent limitations in capturing non-linear relationships and long-term dependencies reduce their applicability to more complex datasets. Gradient boosting algorithms, including XGBoost and LightGBM [60], have demonstrated strong performance in modeling non-linear patterns, achieving high predictive accuracy when supported by robust feature engineering. Nevertheless, these methods lack native mechanisms to model sequential dependencies or long-term temporal patterns, often necessitating labor-intensive preprocessing, such as creating explicit lag features. TinyML [61], a framework designed for deploying machine learning models on highly resource-constrained devices, was initially considered due to its efficiency. However, its limitations in handling complex temporal dependencies and non-linear dynamics made it less suitable for the application. Instead, we selected long short-term memory (LSTM) networks, leveraging the computational capabilities of the Raspberry Pi to deploy these models using TensorFlow. LSTMs are designed explicitly for sequential data, excelling in their ability to model long-term dependencies and non-linear temporal relationships. By utilizing Raspberry Pi's relatively advanced processing power, we avoided the need for additional hardware while achieving robust forecasting performance. This alignment of the LSTM's advanced modeling capabilities with the Raspberry Pi's computational potential underscores the suitability of this approach for addressing complex time series forecasting challenges in embedded systems.

- *Model Development*
  The development of the LSTM-based temperature prediction model followed a systematic process involving data preprocessing, model architecture design, training, evaluation, and deployment. The primary goal was to predict future temperatures based on historical sensor data from multiple nodes while ensuring the model's compatibility with edge devices such as Raspberry Pi. The dataset consisted of over one million temperature readings from 50 sensor nodes stored in a CSV file. The dataset was then split into training, validation, and test sets using an 80-10-10 ratio. The data were normalized to a range of [0, 1] using MinMax scaling to enhance training efficiency.
  Sequences of 20-time steps were generated using a sliding window approach for time series modeling. Each sequence provided the input for the model to predict the temperature for the subsequent hour. This preparation enabled the LSTM model to capture temporal dependencies and patterns within the data.
  The LSTM model architecture was designed to balance predictive accuracy and computational efficiency. The model consisted of two LSTM layers with 50 units capable of processing sequential input and capturing long-term dependencies inherent in the temperature data. Two Dense output layers, one with 50 neurons and one with a single neuron, were used to predict the temperature for the next hour. The input shape for the model was defined as $(n\_steps, n\_features)$, where n_steps represents the number of historical time steps (20) and $n\_features$ corresponds to the number of sensor nodes. The model was trained using the Adam optimizer with a root mean squared error (RMSE) loss function.
  The training process was conducted over 20 epochs with a batch size of 30. Early stopping was implemented with a patience of 5 epochs to mitigate overfitting by halting training when the validation loss ceased to improve. Model performance was evaluated on the test set, with root mean squared error (RMSE) as the primary metric for assessing prediction accuracy, obtaining a result of 0.1838, and a prediction time of approximately 200 ms.
  Following training, the model was prepared for deployment on edge devices. The trained LSTM model was saved in TensorFlow's SavedModel format and converted to TensorFlow Lite format using the TFLiteConverter. The TensorFlow Lite interpreter validated the converted model's performance, ensuring compatibility and consistent predictions.
  This structured approach enabled the LSTM model to leverage the computational capabilities of the Raspberry Pi effectively, providing robust and scalable temperature forecasts while adhering to the constraints of resource-constrained edge environments.
- *The inference code*
  The software for the inference is based on Python 3.11, leveraging libraries such as NumPy 1.26.3, Pandas 2.1.4, uJson 5.4.0, TensorFlow 2.15.0, and Sci-kit learn 1.2.2. Data acquisition begins with readings from a TCP server, which are preprocessed to eliminate erroneous values or measurements outside the acceptable range. Values exceeding this range trigger a warning, which the notification system records. Similarly, if the model predicts a value that is too high or too low, a warning is also triggered to signal potential anomalies. For error values (e.g., NaN), the algorithm ignores them to

prevent distortion in the data. In the event of a disconnection from the TCP server, the program automatically attempts to reconnect after one second. After collecting enough valid measurements, the pre-trained model is loaded, and predictions are generated using a sliding window approach with a span of 20 values. This method ensures that each new measurement dynamically updates the forecast, providing precise and timely insights into sensor behavior while maintaining accuracy and detecting potential anomalies in real-time. To tailor predictions to the unique characteristics of each node, a personalized MinMax Scaler is trained for every sensor. This personalized scaling not only enhances the accuracy of the predictions but also facilitates the monitoring and maintenance of individual nodes, ensuring consistent performance across the network. A flowchart of the algorithm is shown below in Figure 4.

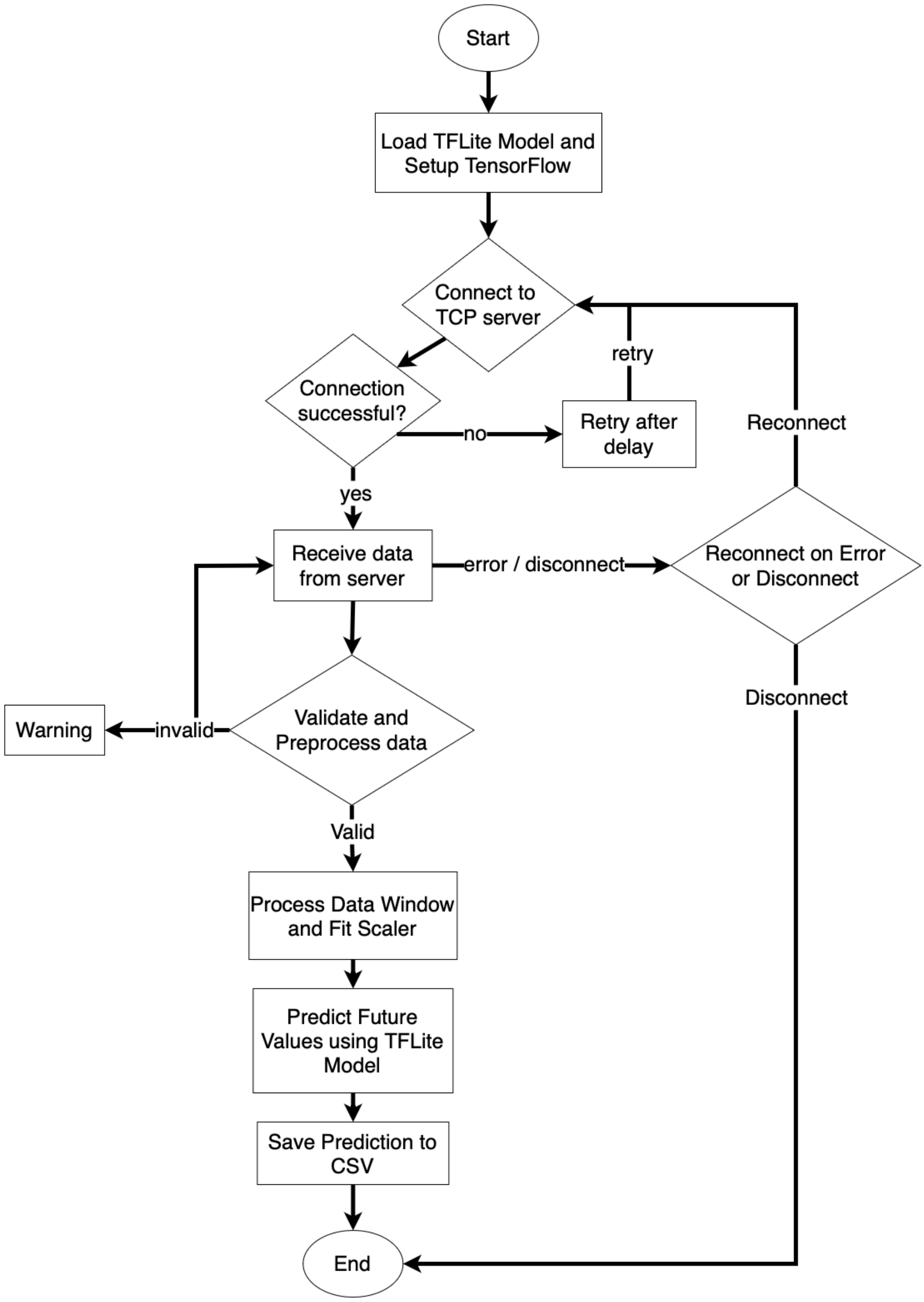

Figure 4. Flowchart of the LSTM algorithm.

- *Requirements of the host system*
  The host system must conduct the following:
  - Provide a sandbox environment for effortless integration of custom algorithms, empowering users to adapt the platform to diverse industrial needs. In our specific case:
    * Receive data from the concentrator via TCP connection and process the raw data files generated by the container.
    * Ensure real-time data processing, perform data cleaning, and apply integrated models or algorithms.
    * Manage a sliding window of 20 values for accurate predictions of future data points.
    * Provide updated predictions with each new data input to enhance monitoring and process optimization.
  - Integrate easily with external systems for data analysis and industrial management.

4. Experiment and Discussion

The experiment consisted of two phases: The first was in a controlled laboratory setting, where 50 sensor nodes were connected to monitor and record the room's temperature. In the second phase, we used an actual machine and tested it in a thermal chamber instead of a real-world environment due to ongoing testing constraints.

The Nicla Sense ME (Arduino, Ivrea, Italy) was used as a sensor node; it exemplifies a trio of strengths: compact design, cost-effectiveness, and energy efficiency, while seamlessly integrating four state-of-the-art sensors engineered by Bosch Sensortec. The acronym "ME" stands for "Motion" and "Environment", highlighting its capability to accurately detect rotation, acceleration, temperature, humidity, pressure, air quality, and CO_2 levels with industrial-grade precision. This latest version sets a new benchmark with its ultra-compact size, enabling effortless data transmission via Bluetooth Low Energy (BLE) 4.2, powered by the ANNA-B112 module. In the experiment, 50 Nicla sensors were used, both in the laboratory and on the machine. Their primary role was to transmit temperature, humidity, and pressure data to the concentrator using the wireless protocol.

The Raspberry Pi 4 Model B (Raspberry Pi Foundation, Cambridge, United Kingdom), with Quad-core Cortex-A72 (ARM v8), was used as a concentrator, the core of our embedded hardware system, operating on Raspberry Pi OS 64-bit, built on Debian 12 (Bookworm). This device provides ample memory (8 GB of RAM) to handle the demands of running resource-intensive applications, including machine learning models and real-time sensor data processing, ensuring smooth performance and efficient multitasking within the system. Regarding power consumption, the system uses embedded devices, such as the Raspberry Pi, which inherently consumes very little energy (typically around 2–5 watts), making it highly suitable for long-term operation in resource-constrained environments. The choice of this hardware for the concentrator and the low-power sensors, such as the Nicla Sense ME, ensures that the system can operate for extended periods without significant energy demands. This efficient power usage is crucial for deployment in remote or industrial environments, where resources must be carefully managed.

To support virtualization and sandboxing for external users, Docker version 25.0.4 (build 1a576c5) was implemented, with Docker Compose version 2.24.7 enabling efficient management of multiple containers concurrently. Given our reliance on Bluetooth for wireless communication, the BlueZ library version 5.66 was integrated into the system. The overall experiment is represented in Figure 5.

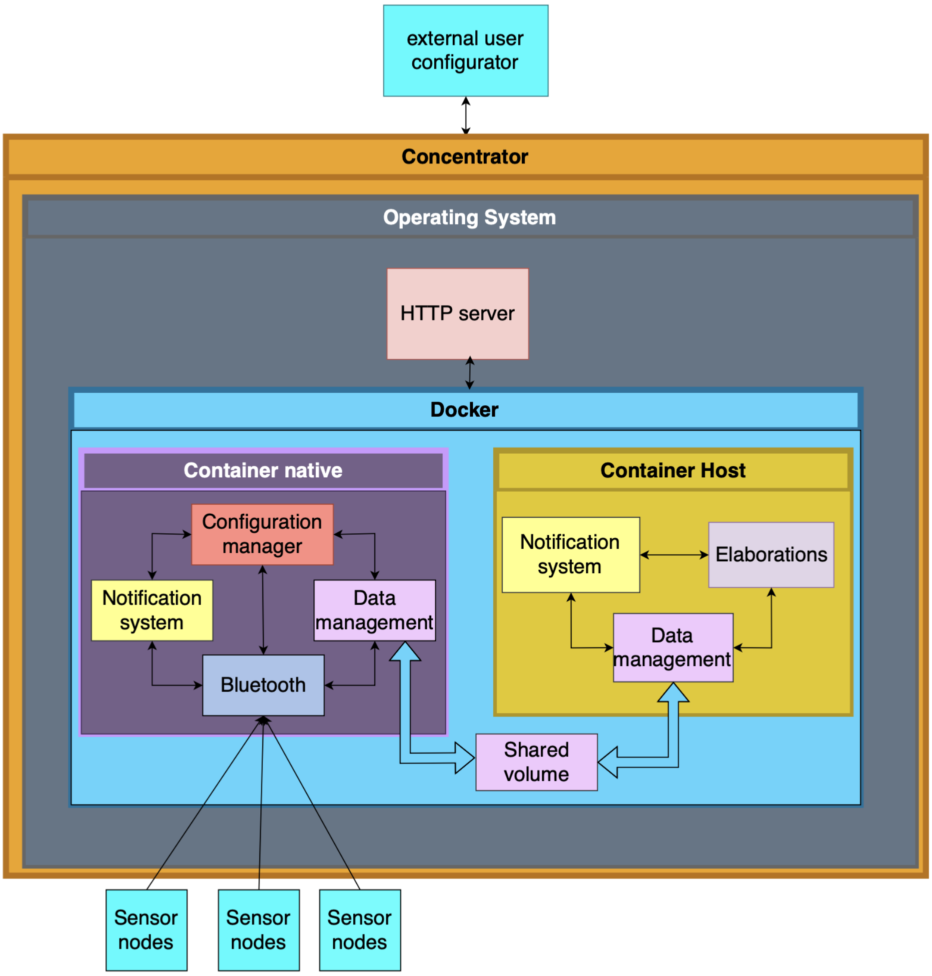

Figure 5. The architecture of the experiment, in which the elaboration part is done by the sandbox (container host).

As represented in Figure 6, the concentrator's primary function is to aggregate data from multiple peripherals. Utilizing a container built on Node.js version 16 and the open-source library node-ble, this system intelligently differentiates between Host and Native configurations based on peripheral addresses and settings. It establishes smooth connections with the targeted peripherals and carefully records the data streams from the Nicla Sense ME sensors. The acquisition frequency is calibrated to capture measurements at a rate of one per second. Once connections are established and the necessary data are acquired, the container publishes the raw data locally using a TCP server. The Native container employs the node-ble library and Bluetooth connections to broadcast native measurements. Each line of the data file includes the peripheral's MAC address, the measured characteristic's value, and the sensor it uses (internal or external, in case an external sensor is attached). The software version for the sensor nodes was v0.6.0 and the Concentrator v0.1.0, both properties of AROL Closure Systems (Canelli, IT).

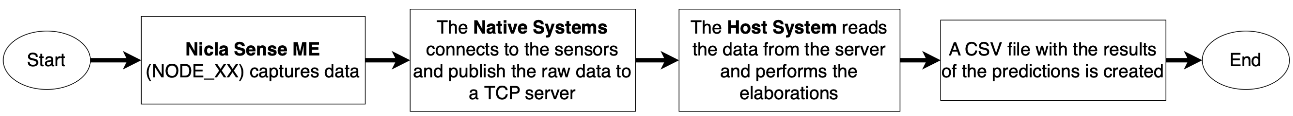

Figure 6. Flowchart of the experiment.

The results are evaluated using the following performance metrics:

- **RMSE (root mean squared error):** This metric quantifies the square root of the average squared differences between the observed and predicted values. It is calculated as follows:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

  where y_i are the observed values, $\hat{y}_i$ are the predicted values, and n is the number of data points.
- **MAE (mean absolute error):** MAE represents the average absolute difference between predicted values and actual values. It is given by the following:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

  where y_i are the observed values, $\hat{y}_i$ are the predicted values, and n is the number of data points.
- **MAPE (mean absolute percentage error):** This metric expresses the average absolute percentage difference between predicted and actual values, offering a relative measure of prediction accuracy. It is calculated as follows:

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100$$

  where y_i are the observed values, $\hat{y}_i$ are the predicted values, and n is the number of data points.
- **Accuracy:** Accuracy is defined as the percentage of correct predictions made by the model out of all predictions. It can be calculated as follows:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{n} \times 100$$

  where the number of correct predictions refers to the instances, where $\hat{y}_i = y_i$, and n is the total number of predictions.
- **R2 (R-squared):** R-squared measures the proportion of variance in the dependent variable that is explained by the independent variables in the model. It is computed as follows:

$$\text{R2} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

  where y_i are the observed values, $\hat{y}_i$ are the predicted values, $\bar{y}$ is the mean of the observed values, and n is the number of data points. A higher R^2 value indicates a better fit of the model to the data.

4.1. Controlled Laboratory Results

The first phase aimed to assess the system's performance and evaluate anomaly detection between the sensors. One was intentionally adjusted within its software to generate more noticeable temperature fluctuations. This sensor was modified to heat up or cool down, creating more pronounced temperature variations than the others and maintaining a uniform reading. These modifications resulted in a change in its prediction without knowing which node had been altered. The Table 1 below presents the resource consumption data for the architecture, showcasing the system's efficiency across various configurations:

The results indicate that the system manages resource consumption effectively as the number of nodes increases. Even with 50 nodes, memory and CPU usage remain within manageable limits, demonstrating the system's scalability without significant performance degradation.

Table 1. LSTM experiment data.

Container	Number of Nodes	Memory	CPU
None	0	570 MB	0%
Only Native	1	671 MB	10%
Only Native	50	700 MB	17%
Only Host	1	800 MB	10%
Only Host	50	1 GB	20%
Docker Compose	1	1.2 GB	35%
Docker Compose	50	1.4 GB	55%

Moreover, the system's transmission efficiency was evaluated as the number of nodes increased. When scaling from a single node (with a detection rate of 60 times per minute) to 50 nodes, the same sensor was detected 44 times per minute, resulting in a 26.53 % decrease in transmission efficiency. Despite this, the overall impact on system performance remains acceptable, suggesting the system is well-suited for environments with multiple nodes, where transmission efficiency is still maintained at a reasonable level.

These findings highlight that the system balances performance, resource usage, and scalability, making it a strong candidate for deployment in industrial IoT environments that require efficient resource management and the ability to scale as needed.

The analysis in Figure 7 demonstrates that minimal error metrics mark the system's predictive performance. The highest values for MAE and RMSE are noted for NODE_36 at 0.0126 and 0.0156, respectively, and an accuracy of 99.81 %. These values represent the most significant deviations among the nodes, yet they still indicate that even the node with the most critical errors remains within a relatively narrow accuracy range. In contrast, NODE_25 is identified as an anomaly within the dataset due to its intentional modification as a test case designed to assess the system's capability of checking the node's behavior. This deliberate alteration of NODE_25 provides insights into the system's behavior under abnormal conditions. The generally low error rates across most nodes highlight the effectiveness of the predictive maintenance system, underscoring its ability to produce accurate and reliable forecasts. The discrepancies observed, especially with NODE_25, illustrate how the concentrator monitors and detects anomalies.

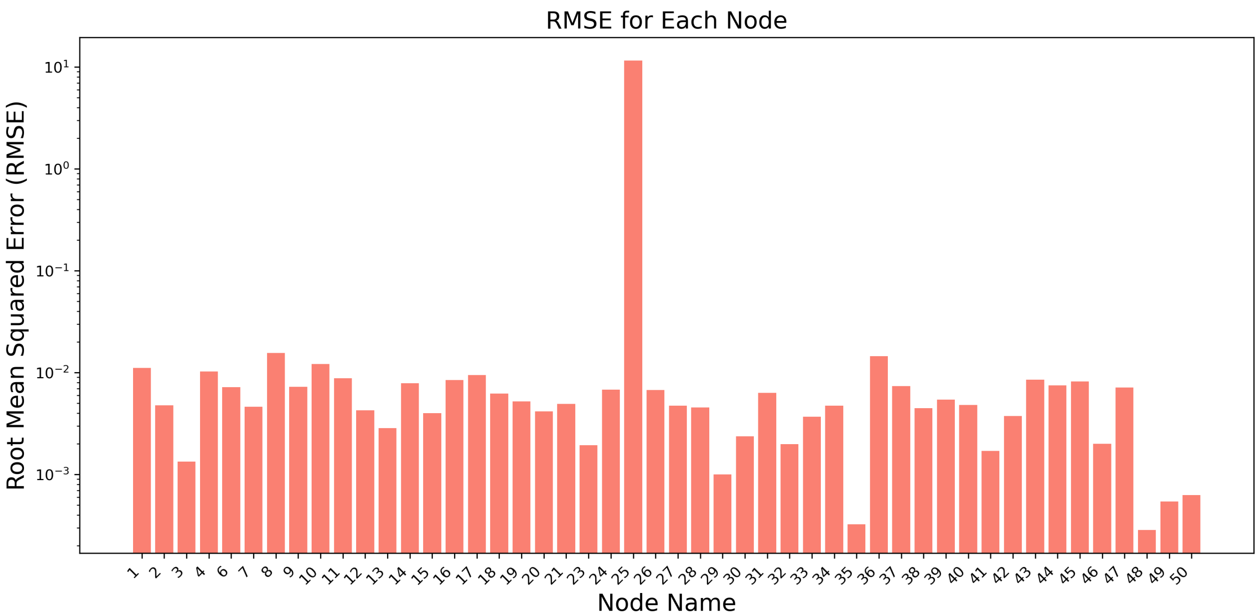

Figure 7. Mean square error of laboratory results (logarithmic scale).

4.2. Real Machine Results

In alignment with laboratory results, the performance metrics obtained from the real machine data, an experiment image is Figure 8, further affirm the system's efficiency, as depicted in Figure 9. Individual sensors were not subject to modification or experimental alteration in the scenario. The consistency is reflected in the results, where all sensors exhibited comparable errors in predictive accuracy and recorded similar temperature measurements. Maximum error values were observed for NODE_08, which recorded a maximum MAE of 0.1821, an RMSE of 0.288, R^2 (R-squared) of 0.8898, a MAPE of 0.7227%, and an accuracy of 99.2% (with an error of $\leq$1 °C), as shown in Figure 10. These values show only minimal deviation, highlighting the system's precision in predicting machine behavior. While the results are less optimal than those from the laboratory tests, this is attributed to the presence of metal and other interferences typical of a real industrial setting. The lack of anomalies or sensor discrepancies in this test led to consistently low error rates, indicating that the system performs optimally when all components operate as expected. This outcome effectively demonstrates the system's ability to maintain high accuracy without external disruptions, further confirming its reliability for practical use in real-world environments.

Table 2 compares the results obtained in the experiments conducted for these experiments.

Table 2. Comparison between laboratory and real application results.

Experiment	Node	MAE	RMSE	Accuracy	R^2	MAPE
Laboratory	NODE_36	0.0126	0.0156	99.81%	0.9970	0.0441
Laboratory	NODE_25 (Anomaly)	2.3526	11.5418	45%	−6,211,580.9	55
Real Machine	NODE_08	0.1821	0.2880	99.2%	0.8898	0.7227

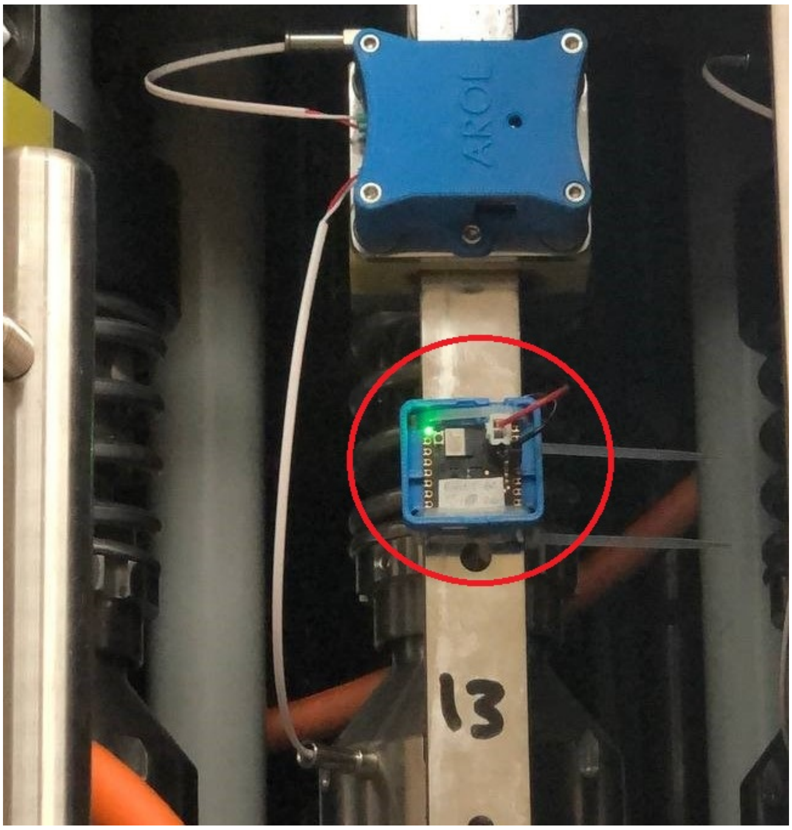

Figure 8. Image of the sensor NODE_13 mounted on the machine, head 13.

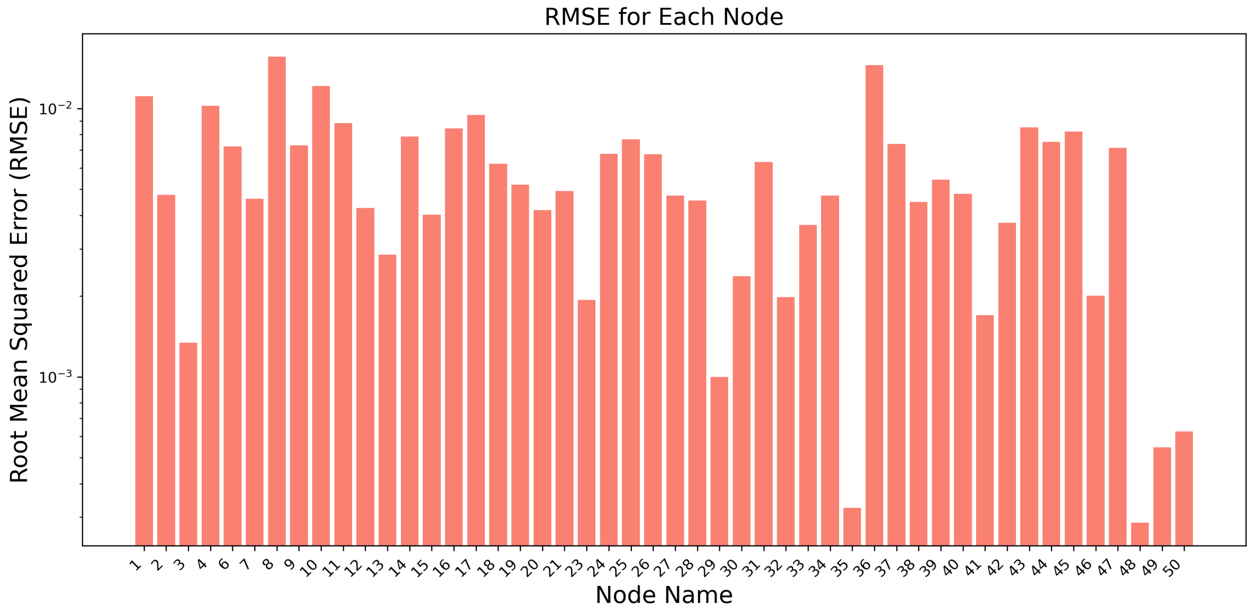

Figure 9. Mean square error of real machine results (logarithmic scale).

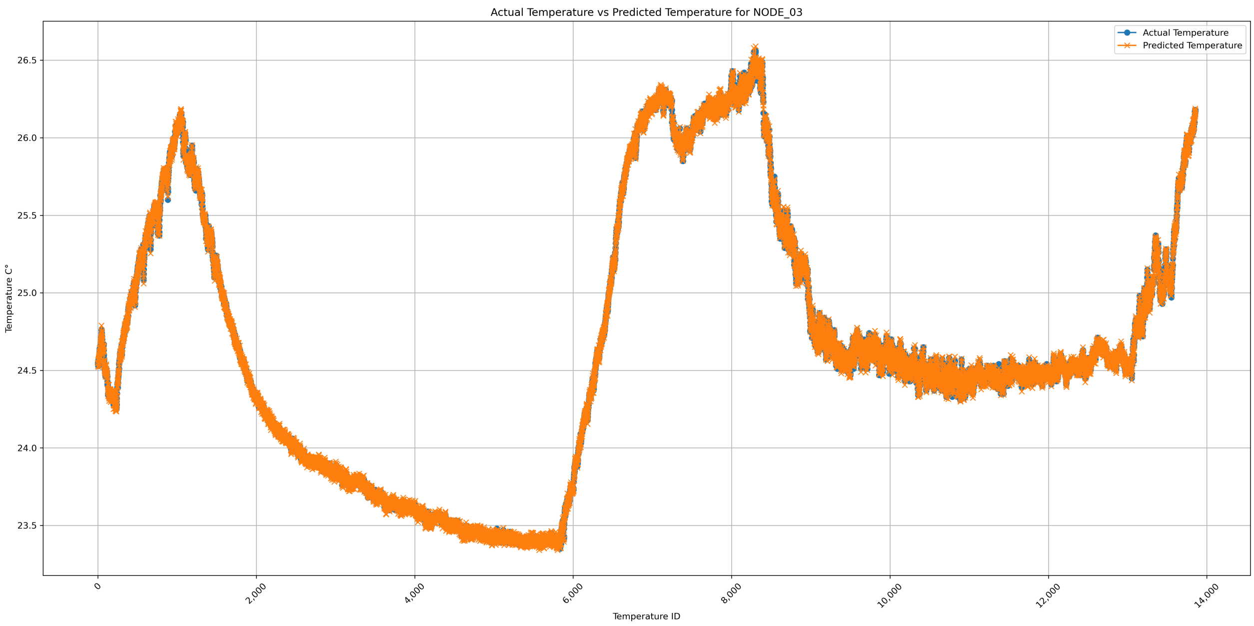

Figure 10. Example of temperatures and their values predicted.

5. Comparison with Existing Approaches

5.1. Comparison with FDPR [44]

Both studies focus on similar use cases involving sensor data prediction and recovery in IoT environments. However, notable differences exist in the models employed, the experimental setups, and the results achieved. Table 3 presents a comparative analysis of key performance metrics between the LSTM-TFLite model (the concentrator) and the model from the previous study (FDPR, DC-MLP). The table highlights significant differences across several performance indicators, including RMSE, MAPE, prediction accuracy, and inference time.

The LSTM-TFLite model demonstrates a higher MAE (0.1821 compared to 0.06) and equal RMSE (0.28 compared to 0.28), suggesting that the DC-MLP model exhibits lower error rates, indicating slightly better performance in error metrics. Additionally, the MAPE for the LSTM-TFLite model (0.7) is significantly larger than the value for the DC-MLP model (0.002), further suggesting that the DC-MLP model offers more accurate predictions relative to the actual values in percentage terms. The DC-MLP model also achieves a higher prediction accuracy (99.86%) compared to the LSTM-TFLite model (99.2%).

The most notable discrepancy is inference time: our model takes 200 ms to process, significantly longer than the previous study's 0.05 ms. This large gap in processing speed could be attributed to differences in the complexity of the models, hardware configurations, or the specific optimization techniques used in each approach. Despite the increased inference time, the overall accuracy of our model and its ability to handle sensor data predictions and recovery are still competitive with the prior work.

Table 3. Comparison of key performance metrics.

Metric	The Concentrator (LSTM-TFLite Model)	FDPR (DC-MLP)
MAE	0.1821	0.06
RMSE	0.28	0.28
MAPE	0.7	0.002
Prediction Accuracy (%)	99.2	99.86
Inference Time (ms)	200	0.05

5.2. Comparison with the FAQMP [45]

Both studies focus on similar use cases involving sensor data prediction and recovery in IoT environments. Table 4 presents a comparative analysis of key performance metrics between the LSTM-TFLite model (the concentrator) and the model from the previous study (FDPR, Seq2Seq GRU Attention-TFLite model).

Table 4. Comparison of key performance metrics.

Metric	The Concentrator (LSTM-TFLite Model)	FAQMP (Seq2Seq GRU Attention-TFLite Model)
MAE	0.1821	4.385
RMSE	0.28	7.9016
MAPE	0.7	22.0133
R2	0.8898	0.6566
Inference Time (ms)	200	52,671.8

The table highlights significant differences across several performance indicators, including MAE, RMSE, mean absolute percentage error (MAPE), R-squared (R2), prediction accuracy, and inference time.

The LSTM-TFLite model demonstrates a significantly lower MAE (0.1821 compared to 4.385) and RMSE (0.28 compared to 7.9016), indicating superior performance regarding error metrics. This result suggests that the LSTM-TFLite model provides more accurate predictions with less deviation from the actual values. Similarly, the MAPE for the LSTM-TFLite model (0.7) is much smaller than that of the Seq2Seq GRU Attention-TFLite model (22.0133), indicating that the LSTM-TFLite model has a lower relative prediction error.

Regarding R-squared (R2), the LSTM-TFLite model also performs better, with an R2 value of 0.8898 compared to the previous study's value of 0.6566. This result indicates that the LSTM-TFLite model explains a higher proportion of the variance in the data, thus providing a better fit and more reliable predictions.

However, the most striking difference is observed in inference time: the LSTM-TFLite model requires 200 ms for inference, while the previous study's model takes an extraordinarily long 52.6 s. This significant discrepancy in processing time may be attributed to the complexity of the Seq2Seq GRU attention model and suggests that the LSTM-TFLite model is much more efficient for real-time applications.

5.3. Comparison with the DeepFogAQ [45]

Both studies focus on similar use cases involving sensor data prediction and recovery in IoT environments. Table 5 presents a comparative analysis of key performance metrics between the LSTM-TFLite model (the concentrator) and the model from the previous study (DeepFogAQ, Transformers).

Table 5. Comparison of key performance metrics.

Metric	The Concentrator (LSTM-TFLite Model)	DeepFogAQ (Transformers)
MAE	0.1821	3.15
RMSE	0.28	3.79

The table highlights significant differences across several performance indicators, including MAE and RMSE.

The LSTM-TFLite model exhibits a lower MAE (0.1821 compared to 3.15) and RMSE (0.28 compared to 3.79), suggesting that it outperforms the DeepFogAQ Transformer model regarding error metrics. This result indicates that the LSTM-TFLite model provides more accurate predictions with less deviation from the actual values.

Despite the differences in error metrics, both models aim to optimize performance for sensor data prediction and recovery. However, the LSTM-TFLite model demonstrates superior prediction accuracy and error reduction, giving it an edge in sensor data prediction tasks.

An overall comparison between all the models can be found in Table 6.

Table 6. Comparison of key performance metrics across different models.

Metric	The Concentrator	FDPR	FAQMP	DeepFogAQ
MAE	0.1821	0.06	4.385	3.15
RMSE	0.28	0.28	7.9016	3.79
MAPE	0.7	0.002	22.0133	-
R2	0.8898	-	0.6566	-
Prediction Accuracy (%)	99.2	99.86	-	-
Inference Time (ms)	200	0.05	52,671.8	-

6. Conclusions and Future Works

This study demonstrated the practicality of implementing a predictive maintenance system within a resource-constrained IoT framework. The system successfully monitored and predicted real-time temperature measurements by utilizing Nicla Sense ME sensors and a Raspberry Pi-based concentrator. Including a sandbox environment allows users to customize the system by adding different codes or sensors without modifying the core software, enhancing its flexibility for various applications. However, as device numbers and data volume increase, managing scalability may require further optimization to handle the growing data flow efficiently.

Implementing an LSTM-based machine learning model on an embedded system showcased how advanced AI techniques can be applied effectively in environments with limited computational resources. The model's precise predictions, with an RMSE of 0.1838, highlight its potential for industrial applications aimed at reducing downtime and enabling proactive maintenance. As the system scales, though, the computational load could strain embedded devices, calling for additional optimization or alternative hardware solutions.

The system's scalability was further validated as it maintained reliable performance across multiple connected devices. This demonstrates its potential for broader deployment in smart industrial environments. However, ensuring compatibility with diverse industrial machines and sensors that use proprietary protocols may present challenges, especially when managing real-time data transmission across potentially congested networks.

Future enhancements will focus on improving cybersecurity to mitigate emerging threats, an essential step as IoT systems become complex. Integrating data from additional machine parameters, such as accelerations and vibrations, will offer a more detailed understanding of machine behavior, further enhancing predictive maintenance capabilities.

The system will also undergo rigorous stress testing to evaluate its resilience in high-data-load scenarios, network disruptions, and extreme environmental conditions. These tests will provide insights into its performance and reliability in real-world industrial settings, helping to identify potential weaknesses such as hardware degradation or sensitivity to environmental stresses. These improvements will not only bolster the system's security but also enhance its robustness, ensuring more accurate maintenance strategies moving forward.

Author Contributions: Conceptualization, P.D.; Methodology, P.D.; Software, P.D.; Validation, P.D.; Formal analysis, P.D.; Investigation, P.D.; Resources, P.D.; Data curation, P.D.; Writing—original draft, P.D.; Writing—review & editing, P.D., M.V. and G.M.; Visualization, P.D.; Supervision, P.D.; Project administration, P.D.; Funding acquisition, P.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: The authors express their deepest gratitude to AROL Closure Systems (Canelli, IT) for generously sharing their expertise and resources, which were instrumental in completing this project. Furthermore, they sincerely thank their colleagues, whose invaluable assistance significantly contributed to the development of this paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Saeed, A.; Khattak, M.A.K.; Rashid, S. Role of Big Data Analytics and Edge Computing in Modern IoT Applications: A Systematic Literature Review. In Proceedings of the 2022 2nd International Conference on Digital Futures and Transformative Technologies (ICoDT2), Rawalpindi, Pakistan, 24–26 May 2022; pp. 1–5.
2. Mathur, S.; Verma, A.; Srivastav, G. Analysis of the Three Layers of Computing: Cloud, Fog, and Edge. In Proceedings of the 2023 International Conference on Computational Intelligence, Communication Technology and Networking (CICTN), Ghaziabad, India, 20–21 April 2023; pp. 463–466.
3. Forsström, S.; Lindqvist, H. Evaluating Scalable Work Distribution Using IoT Devices in Fog Computing Scenarios. In Proceedings of the 2023 IEEE 9th World Forum on Internet of Things (WF-IoT), Aveiro, Portugal, 12–27 October 2023; pp. 1–6.
4. Khan, Z.A.; Aziz, I.A.; Osman, N.A.B.; Ullah, I. A Review on Task Scheduling Techniques in Cloud and Fog Computing: Taxonomy, Tools, Open Issues, Challenges, and Future Directions. *IEEE Access* **2023**, *11*, 143417–143445. [CrossRef]
5. Tianfield, H. Towards Edge-Cloud Computing. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10–13 December 2018; pp. 4883–4885.
6. Rajanikanth, A. Fog Computing: Applications, Challenges, and Opportunities. *J. Auton. Intell.* **2023**, *24*, 5.
7. Atlam, H.F.; Walters, R.J.; Wills, G.B. Fog Computing and the Internet of Things: A Review. *Big Data Cogn. Comput.* **2018**, *2*, 10. [CrossRef]
8. Mouradian, C.; Naboulsi, D.; Yangui, S.; Glitho, R.; Morrow, M.; Polakos, P. A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges. *IEEE Commun. Surv. Tutor.* **2017**, *20*, 416–464. [CrossRef]
9. Abdali, T.-A.N.; Hassan, R.; Aman, A.H.M.; Nguyen, Q.N. Fog Computing Advancement: Concept, Architecture, Applications, Advantages, and Open Issues. *IEEE Access* **2021**, *9*, 75961–75980. [CrossRef]

10. Singh, J.; Singh, P.; Gill, S.S. Fog Computing: A Taxonomy, Systematic Review, Current Trends and Research Challenges. *J. Parallel Distrib. Comput.* **2021**, *157*, 6. [CrossRef]
11. Weng, C.-Y.; Li, C.-T.; Chen, C.-L.; Lee, C.-C.; Deng, Y.-Y. A Lightweight Anonymous Authentication and Secure Communication Scheme for Fog Computing Services. *IEEE Access* **2021**, *9*, 145522–145537. [CrossRef]
12. Rana, P.; Walia, K.; Kaur, A. Challenges in Conglomerating Fog Computing with IoT for Building Smart City. In Proceedings of the International Conference on Computational Methods in Science & Technology (ICCMST), Mohali, India, 17–18 December 2021; Volume 2, pp. 35–40.
13. Kumar, K.V.; Kumar, K.; Kumar, R.; Basha, S.M.; Praveen, M.; Reddy, P. Internet of Things and Fog Computing Applications in Intelligent Transportation Systems. In *Architecture and Security Issues in Fog Computing Applications*; IGI Global: Hershey, PA, USA, 2019; Chapter 008.
14. Stephen, V.K.; Udinookkaran, P.; De Vera, R.P.; Al-Harthy, F.R.A. Internet of Medical Things (IoMT) Fog-Based Smart Health Monitoring. In Proceedings of the 2023 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), Chennai, India, 25–26 May 2023; pp. 1–8.
15. Rasheed, M.; Saleem, J.; Hudabia, M.; Hafiz, T.; Mannan, R.; Mishaal, A. A Survey on Fog Computing in IoT. *VFAST Trans. Softw. Eng.* **2021**, *9*, 4. [CrossRef]
16. *IEEE Std 2672-2023*; IEEE Guide for General Requirements of Mass Customization. IEEE: Piscataway, NJ, USA, 2023; pp. 1–52.
17. Ahmed, K.; Zeebaree, S. Resource Allocation in Fog Computing: A Review. In Proceedings of the 2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC), Paris, France, 20–23 April 2020; Volume 10, p. 5.
18. Mukherjee, M.; Matam, R.; Shu, L.; Maglaras, L.; Ferrag, M.A.; Choudhury, N.; Kumar, V. Security and Privacy in Fog Computing: Challenges. *IEEE Access* **2017**, 5, 19293–19304. [CrossRef]
19. Kumar, S.; Dwivedi, M.; Kumar, M.; Gill, S.S. A Comprehensive Review of Vulnerabilities and AI-Enabled Defense against DDoS Attacks for Securing Cloud Services. *Comput. Sci. Rev.* **2023**, *53*, 100661. [CrossRef]
20. Kolevski, D.; Michael, K. Edge Computing and IoT Data Breaches: Security, Privacy, Trust, and Regulation. *IEEE Technol. Soc. Mag.* **2024**, *43*, 22–32. [CrossRef]
21. Pakmehr, A.; Aßmuth, A.; Taheri, N.; Ghaffari, A. DDoS Attack Detection Techniques in IoT Networks: A Survey. *Clust. Comput.* **2024**, 27, 14637–14668. [CrossRef]
22. Alsadie, D. Artificial Intelligence Techniques for Securing Fog Computing Environments: Trends, Challenges, and Future Directions. *IEEE Access* **2024**, *12*, 151598–151648. [CrossRef]
23. Marchang, J.; McDonald, J.; Keishing, S.; Zoughalian, K.; Mawanda, R.; Delhon-Bugard, C.; Bouillet, N.; Sanders, B. Secure-by-Design Real-Time Internet of Medical Things Architecture: E-Health Population Monitoring (RTPM). *Telecom* **2024**, *5*, 609–631. [CrossRef]
24. Kumar, R.; Singh, S.; Singh, D.; Kumar, M.; Gill, S.S. A Robust and Secure User Authentication Scheme Based on Multifactor and Multi-Gateway in IoT-Enabled Sensor Networks. *Secur. Priv.* **2024**, *7*, e335. [CrossRef]
25. Shukla, P.; Krishna, C.R.; Patil, N.V. IoT Traffic-Based DDoS Attacks Detection Mechanisms: A Comprehensive Review. *J. Supercomput.* **2024**, 80, 9986–10043. [CrossRef]
26. d'Agostino, P.; Violante, M.; Macario, G. A User-Extensible Solution for Deploying Fog Computing in Industrial Applications. In Proceedings of the 2023 IEEE 32nd International Symposium on Industrial Electronics (ISIE), Helsinki, Finland, 19–21 June 2023; pp. 1–6.
27. Fava, F.B.; Leite, L.F.L.; Da Silva, L.F.A.; Costa, P.R.D.S.A.; Nogueira, A.G.D.; Lopes, A.F.G.; Mansilha, R.B. Assessing the Performance of Docker in Docker Containers for Microservice-Based Architectures. In Proceedings of the 2024 32nd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Dublin, Ireland, 20–22 March 2024; pp. 137–142.
28. D'Agostino, P.; Violante, M.; Macario, G. An Embedded Low-Cost Solution for a Fog Computing Device on the Internet of Things. In Proceedings of the 2023 Eighth International Conference on Fog and Mobile Edge Computing (FMEC), Tartu, Estonia, 18–20 September 2023; pp. 284–291.
29. D'Agostino, P.; Violante, M.; Macario, G. Optimizing LSTM-Based Temperature Prediction Algorithm for Embedded System Deployment. In Proceedings of the Emerging Technologies and Factory Automation (ETFA), Padova, Italy, 10–13 September 2024; pp. 1–7.
30. Masini, R.P.; Medeiros, M.C.; Mendes, E.F. Machine Learning Advances for Time Series Forecasting. *J. Econ. Surv.* **2023**, *37*, 76–111. [CrossRef]
31. Li, G.; Jung, J.J. Deep Learning for Anomaly Detection in Multivariate Time Series: Approaches, Applications, and Challenges. *Inf. Fusion* **2023**, *81*, 1–14. [CrossRef]
32. Zhang, C.; Yang, Z.; Zhang, H.; Yao, L.; Xie, Y.; Sui, J.; Liao, Q. Deep-learning Approach for Predicting Performance in Fog Computing Systems. *IEEE Trans. Ind. Inform.* **2024**, *20*, 2.
33. Lera, I.; Guerrero, C.; Juiz, C. YAFS: A Simulator for IoT Scenarios in Fog Computing. *IEEE Access* **2019**, *7*, 91745–91758. [CrossRef]

34. Wöbker, C.; Seitz, A.; Mueller, H.; Bruegge, B. Fogernetes: Deployment and Management of Fog Computing Applications. In Proceedings of the 2018 IEEE/IFIP Network Operations and Management Symposium (NOMS), Taipei, Taiwan, 23–27 April 2018; pp. 1–7.
35. D'Alessandro, D.; Gunderson, W.; Staten, E.; Donastien, Y.K.; Rodriguez, P.; Bailey, R. Integrating Modularity for Mass Customization of IoT Wireless Sensor Systems. In Proceedings of the 2021 Systems and Information Engineering Design Symposium (SIEDS), Virtual Conference, 29–30 April 2021; pp. 1–5.
36. Mahmud, R.; Toosi, A.N. Con-Pi: A Distributed Container-Based Edge and Fog Computing Framework. *IEEE Internet Things J.* **2022**, *9*, 4125–4138. [CrossRef]
37. Samaras, T.; Dimara, A.; Tzallas, P.; Papaioannou, A.; Bezas, N.; Krinidis, S.; Anagnostopoulos, C.; Ioannidis, D.; Tzovaras, D. Edge-Computing FogFlow Framework For Solar Generation Prediction Exploiting Federated Learning. In Proceedings of the 2023 International Conference on Modeling & E-Information Research, Artificial Learning and Digital Applications (ICMERALDA), Karawang, Indonesia, 24 November 2023; pp. 149–154.
38. Li, H.; Zhou, S.; Yuan, B.; Zhang, M. Optimizing Intelligent Edge Computing Resource Scheduling Based on Federated Learning. *J. Knowl. Learn. Sci. Technol.* **2024**, *3*, 235–260. [CrossRef]
39. Jadhav, A.; Gaikwad, R.; Patekar, T.; Dighe, S.; Shaikh, B.; Patankar, N.S. Predictive Maintenance of Industrial Equipment Using IoT and Machine Learning. In Proceedings of the 2023 4th International Conference on Computation, Automation and Knowledge Management (ICCAKM), Dubai, United Arab Emirates, 12–13 December 2023; pp. 1–5.
40. Dhanraj, D.; Sharma, A.; Kaur, G.; Mishra, S.; Naik, P.; Singh, A. Comparison of Different Machine Learning Algorithms for Predictive Maintenance. In Proceedings of the 2023 International Conference for Advancement in Technology (ICONAT), Goa, India, 24–26 January 2023; pp. 1–7.
41. Amer, S.; Mohamed, H.K.; Badr Monir Mansour, M. Predictive Maintenance by Machine Learning Methods. In Proceedings of the 2023 Eleventh International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, Egypt, 21–23 November 2023; pp. 58–66.
42. Laganà, F.; Pratticò, D.; Angiulli, G.; Oliva, G.; Pullano, S.A.; Versaci, M.; La Foresta, F. Development of an Integrated System of sEMG Signal Acquisition, Processing, and Analysis with AI Techniques. *Signals* **2024**, *5*, 476–493. [CrossRef]
43. Jouini, O.; Sethom, K.; Namoun, A.; Aljohani, N.; Alanazi, M.H.; Alanazi, M.N. A Survey of Machine Learning in Edge Computing: Techniques, Frameworks, Applications, Issues, and Research Directions. *Technologies* **2024**, *12*, 81. [CrossRef]
44. Trilles, S.; Hammad, S.S.; Iskandaryan, D. Anomaly detection based on Artificial Intelligence of Things: A Systematic Literature Mapping. *Internet Things* **2024**, *25*, 101063. [CrossRef]
45. Abdulazeez, D.H.; Askar, S.K. Offloading Mechanisms Based on Reinforcement Learning and Deep Learning Algorithms in the Fog Computing Environment. *IEEE Access* **2023**, *11*, 12555–12586. [CrossRef]
46. Putra, M.A.P.; Hermawan, A.P.; Nwakanma, C.I.; Kim, D.-S.; Lee, J.-M. FDPR: A Novel Fog Data Prediction and Recovery Using Efficient DL in IoT Networks. *IEEE Internet Things J.* **2023**, *10*, 16895–16906. [CrossRef]
47. Pazhanivel, D.B.; Velu, A.N.; Palaniappan, B.S. Design and Enhancement of a Fog-Enabled Air Quality Monitoring and Prediction System: An Optimized Lightweight Deep Learning Model for a Smart Fog Environmental Gateway. *Sensors* **2024**, 24, 5069. [CrossRef] [PubMed]
48. Şimsek, M.U.; Kök, İ.; Özdemir, S. DeepFogAQ: A fog-assisted decentralized air quality prediction and event detection system. *Expert Syst. Appl.* **2024**, *251*, 123920. [CrossRef]
49. Liu, Y.; Zhao, J.; Wang, L.; Wang, W. Unified Modeling for Multiple-Energy Coupling Device of Industrial Integrated Energy System. *IEEE Trans. Ind. Electron.* **2023**, *70*, 1005–1015. [CrossRef]
50. Xu, Q.; Chen, Z.; Wu, K.; Wang, C.; Wu, M.; Li, X. KDnet-RUL: A Knowledge Distillation Framework to Compress Deep Neural Networks for Machine Remaining Useful Life Prediction. *IEEE Trans. Ind. Electron.* **2022**, *69*, 2022–2032. [CrossRef]
51. Alam, S.; Yakopcic, C.; Wu, Q.; Barnell, M.; Khan, S.; Taha, T.M. Survey of Deep Learning Accelerators for Edge and Emerging Computing. *Electronics* **2024**, *13*, 2988. [CrossRef]
52. Yu, Y.; Guo, L.; Gao, H.; He, Y.; You, Z.; Duan, A. FedCAE: A New Federated Learning Framework for Edge-Cloud Collaboration Based Machine Fault Diagnosis. *IEEE Trans. Ind. Electron.* **2024**, *71*, 4108–4119. [CrossRef]
53. Tufail, S.; Riggs, H.; Tariq, M.; Sarwat, A.I. Advancements and Challenges in Machine Learning: A Comprehensive Review of Models, Libraries, Applications, and Algorithms. *Electronics* **2023**, *12*, 1789. [CrossRef]
54. Ayankoso, S.; Olejnik, P. Time-Series Machine Learning Techniques for Modeling and Identification of Mechatronic Systems with Friction: A Review and Real Application. *Electronics* **2023**, *12*, 3669. [CrossRef]
55. Ang, Y.; Huang, Q.; Bao, Y.; Tung, A.K.; Huang, Z. TSGBench: Time Series Generation Benchmark. *arXiv* **2023**, arXiv:2309.03755. [CrossRef]
56. Sherstinsky, A. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Phys. D Nonlinear Phenom.* **2020**, *404*, 132306. [CrossRef]

57. Al-Masri, E.; Kalyanam, K.R.; Batts, J.; Kim, J.; Singh, S.; Vo, T.; Yan, C. Investigating Messaging Protocols for the Internet of Things (IoT). *IEEE Access* **2020**, *8*, 94880–94911. [CrossRef]
58. Shumway, R.H.; Stoffer, D.S. *ARIMA Models, Time Series Analysis and Its Applications: With R Examples*; Springer International Publishing: Berlin/Heidelberg, Germany, 2017; pp. 75–163.
59. Anand, A.; Srivastava, D.; Rani, L. Anomaly Detection and Time Series Analysis. In Proceedings of the 2023 International Conference on IoT, Communication and Automation Technology (ICICAT), Gorakhpur, India, 23–24 June 2023; pp. 1–5.
60. Aslam, F. Advancing Credit Card Fraud Detection: A Review of Machine Learning Algorithms and the Power of Light Gradient Boosting. *Am. J. Comput. Sci. Technol.* **2024**, *7*, 9–12. [CrossRef]
61. Elhanashi, A.; Dini, P.; Saponara, S.; Zheng, Q. Advancements in TinyML: Applications, Limitations, and Impact on IoT Devices. *Electronics* **2024**, *13*, 3562. [CrossRef]