future internet

MDPI

Article

TinyML Algorithms for Big Data Management in Large-Scale IoT Systems

Aristeidis Karras [1,*], Anastasios Giannaros [1], Christos Karras [1,*], Leonidas Theodorakopoulos [2], Constantinos S. Mammassis [3], George A. Krimpas [1] and Spyros Sioutas [1]

[1] Computer Engineering and Informatics Department, University of Patras, 26504 Patras, Greece; giannaros@ceid.upatras.gr (A.G.); krimpas@upatras.gr (G.A.K.); sioutas@ceid.upatras.gr (S.S.)
[2] Department of Management Science and Technology, University of Patras, 26334 Patras, Greece; theodleo@upatras.gr
[3] Department of Industrial Management and Technology, University of Piraeus, 18534 Piraeus, Greece; cmam@unipi.gr
* Correspondence: akarras@ceid.upatras.gr (A.K.); c.karras@ceid.upatras.gr (C.K.)

Abstract: In the context of the Internet of Things (IoT), Tiny Machine Learning (TinyML) and Big Data, enhanced by Edge Artificial Intelligence, are essential for effectively managing the extensive data produced by numerous connected devices. Our study introduces a set of TinyML algorithms designed and developed to improve Big Data management in large-scale IoT systems. These algorithms, named TinyCleanEDF, EdgeClusterML, CompressEdgeML, CacheEdgeML, and TinyHybridSenseQ, operate together to enhance data processing, storage, and quality control in IoT networks, utilizing the capabilities of Edge AI. In particular, TinyCleanEDF applies federated learning for Edge-based data cleaning and anomaly detection. EdgeClusterML combines reinforcement learning with self-organizing maps for effective data clustering. CompressEdgeML uses neural networks for adaptive data compression. CacheEdgeML employs predictive analytics for smart data caching, and TinyHybridSenseQ concentrates on data quality evaluation and hybrid storage strategies. Our experimental evaluation of the proposed techniques includes executing all the algorithms in various numbers of Raspberry Pi devices ranging from one to ten. The experimental results are promising as we outperform similar methods across various evaluation metrics. Ultimately, we anticipate that the proposed algorithms offer a comprehensive and efficient approach to managing the complexities of IoT, Big Data, and Edge AI.

Keywords: TinyML; Edge AI; IoT; IoT data engineering; IoT Big Data management; IoT systems

Citation: Karras, A.; Giannaros, A.; Karras, C.; Theodorakopoulos, L.; Mammassis, C.S.; Krimpas, G.A.; Sioutas, S. TinyML Algorithms for Big Data Management in Large-Scale IoT Systems. *Future Internet* **2024**, *16*, 42. https://doi.org/10.3390/fi16020042

Academic Editor: Iwona Grobelna

Received: 6 December 2023
Revised: 22 January 2024
Accepted: 23 January 2024
Published: 25 January 2024

1. Introduction

The rapid evolution of the Internet of Things (IoT) has significantly influenced various sectors, including supply chains, healthcare, and energy systems. This technology's expansive data generation presents challenges in managing and interpreting vast volumes of information. Tiny Machine Learning (TinyML), combining the efficiency of embedded systems with advanced machine learning techniques, stands out as a critical solution for facilitating effective, localized data processing within the IoT framework.

The research underscores the growing impact of IoT technologies like TinyML across multiple industries. In supply chain management, IoT's role is instrumental in enhancing inventory management, marking a crucial shift towards Industry 4.0 technologies and their potential to add value to businesses [1]. The healthcare sector benefits from IoT in enhancing services such as remote patient monitoring and real-time data analytics, albeit facing challenges in handling the growth in medical data [2]. TinyML, in particular, is recognized for initiating a new era of IoT and autonomous system applications, attributed to its power efficiency, enhanced privacy, and reduced latency in data processing [3].

The application of machine learning algorithms in embedded systems, including TinyML, is being extensively researched. Studies focus on analyzing performance and developing intelligent systems like solar collectors for thermal energy storage [4,5]. Moreover, integrating multiple blockchain technologies [6,7] and decentralized authentication systems [8] has been proposed as a viable strategy for secure and efficient IoT data management, addressing the digital challenges in the Internet of Everything era [9].

From an industrial perspective, effectively managing TinyML at scale, particularly with regard to the inherent hardware and software constraints of IoT devices, remains a focal area of ongoing research. Proposals suggest frameworks that utilize Semantic Web technologies for the integrated management of TinyML models and IoT devices [10]. Additionally, the advancement and implementation of sophisticated intelligent-anomaly-based intrusion detection systems within IoT networks, utilizing advanced machine learning models, have demonstrated significant efficacy in precisely detecting and proactively mitigating malicious cyber activities [11]. This approach is crucial for enhancing network security and ensuring the reliability of IoT systems in various industrial applications.

The impact of TinyML extends to many areas, including wearable technology [12–16], smart cities [17–22], smart homes [23–25], smart agriculture [26–31], climatic change, environment protection, green AI sustainable applications [32–38], and automobiles [39,40]. Overcoming TinyML's challenges, especially in hardware, is key and can be advanced through creating an open-source community. This initiative would support system development, enhance learning capabilities on small-scale hardware, and help unify software across different platforms [41]. TinyML is set to not only meet current technological demands but also shape the future of smart, efficient data processing.

However, deploying TinyML on microcontrollers presents certain challenges. These include the selection of appropriate programming languages, limited support for various development boards, often overlooked preprocessing steps, the choice of suitable sensors, and a scarcity of labeled data for training purposes [42]. Overcoming these obstacles is essential for the development of TinyML systems that are both efficient and effective.

Tiny Machine Learning (TinyML) offers significant progress in machine learning, focusing on low-resource embedded devices [43]. Despite its extensive potential for enabling machine learning in compact formats, TinyML encounters several challenges, including the absence of standardized benchmarks, limited development board support, programming language restrictions, preprocessing oversights, sensor selection issues, and a lack of sufficient labeled data [42,44]. In regions like Africa, where the adoption of AI and embedded systems is still in its infancy, TinyML could address issues related to connectivity, energy, and costs [45]. Moreover, its ability to facilitate independent operation from cloud computing opens new areas for energy-efficient, secure, and private data processing [46], introducing a new era of novel localized applications in various fields.

In the vast ecosystem of Big Data, TinyML stands out as a significant opportunity. Given the immense volume, velocity, and variety of data in modern systems, traditional Big Data management methods often reach their limits. TinyML presents an innovative solution, functioning more than simply as a tool for data processing but also as an effective technique for data management. By enabling devices to conduct initial data processing and improvement at the network's edge, TinyML significantly reduces the amount of data requiring centralized processing. The subsequent selective transmission and storage of critical data enhance both storage and processing efficiency. Moreover, this approach facilitates quick and immediate analytics, thereby enhancing the overall value gained from Big Data.

The aim of this study focuses on the field of IoT data management by introducing and comparing a suite of specialized TinyML algorithms. Each algorithm, namely TinyCleanEDF (Algorithm 1), EdgeClusterML (Algorithm 2), CompressEdgeML (Algorithm 3), CacheEdgeML (Algorithm 4), and TinyHybridSenseQ (Algorithm 5), has been designed to address specific challenges inherent in IoT systems. TinyML algorithms are essential in this context as they enable efficient processing and intelligent decisionmaking directly

on IoT devices, reducing the reliance on central systems and minimizing data transmission needs. By incorporating advanced techniques such as federated learning, anomaly detection, adaptive data compression, strategic caching, and detailed data quality assessment, these algorithms jointly enhance the overall efficiency, security, and reliability of data management within IoT networks. The comparative analysis provided in this study underscores the distinct functionalities and advantages of each algorithm, highlighting the necessity and versatility of TinyML in handling data in the increasingly complex landscape of IoT systems.

The remainder of this study is organized as follows: Section 2 provides comprehensive background on TinyML, highlighting its emergence as a pivotal tool in managing Big Data within IoT environments and discussing its application in large-scale IoT systems and embedded devices. Section 3 outlines our methodology, covering our approach's advantages, framework design, hardware setup, and dataset configuration for TinyML evaluation. Section 3.7 details the proposed algorithms we developed to utilize TinyML in IoT contexts. The experimental results of the proposed algorithms are thoroughly presented in Section 4, demonstrating the practical implications and effectiveness of our approach. Finally, the study concludes in Section 5, summarizing the key findings and discussing future research directions, emphasizing TinyML's impact on IoT data management.

2. Background and Related Work

IoT systems frequently generate vast quantities of data, posing substantial management and analysis challenges. Researchers have introduced several frameworks and architectures to address these challenges in IoT Big Data management and knowledge extraction. One such proposal is the Cognitive-Oriented IoT Big Data Framework (COIB Framework), as outlined in Mishra's works [47,48]. This framework encompasses an implementation architecture, layers for IoT Big Data, and a structure for organizing data. An alternative method involves employing a Big-Data-enhanced system, adhering to a data lake architecture [49]. Key features of this system include a multi-threaded parallel approach for data ingestion, strategies for storing both raw and processed IoT data, a distributed cache layer, and a unified SQL-based interface for exploring IoT data. Furthermore, blockchain technologies have been investigated for their potential to maintain continuous integrity in IoT Big Data management [50]. This involves five integrity protocols implemented across three stages of IoT operations.

2.1. Big Data Challenges, Internet of Things, and TinyML

The rapid expansion of the Internet of Things (IoT) marks a significant shift in the digital landscape, marked by an extensive network of devices and sensors continuously collecting and sending data. This fusion of environments rich in data greatly increases the challenges related to Big Data, particularly concerning its large volume, high speed, and varied complexity.

The Big Data Dilemma in IoT

As IoT systems evolve, they inherently generate data that challenge conventional processing and storage infrastructures. Key challenges arising from this scenario include:

- **Storage Capacity and Scalability**: Traditional storage systems grapple with the ever-growing influx of data from IoT sources, necessitating the development of more scalable and adaptive solutions.
- **Data Processing and Analysis**: The heterogeneity of IoT data requires sophisticated adaptable algorithms and infrastructures to derive meaningful insights efficiently.
- **Data Transfer and Network Load**: Ensuring efficient and timely data transmission across a myriad of devices without overburdening the network infrastructure remains a paramount concern.
- **Data Integrity and Security**: As data become increasingly decentralized across devices, ensuring their authenticity and safeguarding them from potential threats are critical.

These challenges, which highlight the wider complexities that IoT introduces to Big Data management, are outlined in Table 1.

Table 1. Overview of Big Data challenges in IoT.

Challenge	Description
Data Volume	The increase in interconnected devices leads to unprecedented data generation, surpassing the capacity of conventional storage and processing systems.
Data Velocity	Continuous data generation in IoT necessitates real-time analysis and response, stressing the need for prompt processing solutions.
Data Variety	Diverse data sources in IoT range from structured to unstructured formats, posing integration and analytical complexities.
Data Veracity	The accuracy, authenticity, and reliability of data from varied devices present significant challenges in data verification.
Data Integration	Consolidating data from heterogeneous sources while preserving integrity and context remains complex.
Security	Increased interconnectivity broadens the risk of cyber attacks, necessitating robust security protocols.
Privacy	Balancing the protection of sensitive data within extensive datasets, while maintaining utility, is crucial.
Latency	Processing or transmission delays can affect the timeliness and relevance of insights, impacting decisionmaking.

2.2. TinyML

Tiny Machine Learning (TinyML) has emerged as a growing field in machine learning, characterized by its application in highly constrained Internet of Things (IoT) devices such as microcontrollers (MCUs) [51]. This technology facilitates the use of deep learning models across a multitude of IoT devices, thereby broadening the range of potential applications and enabling ubiquitous computational intelligence. The implementation of TinyML is challenging, primarily due to the limited memory resources of these devices and the necessity for simultaneous algorithm and system stack design. Attracting substantial interest in both research and development areas, numerous studies have been conducted, focusing on the challenges, applications, and advantages of TinyML [52,53].

An essential goal of TinyML is to bring machine learning capabilities to battery-powered intelligent devices, allowing them to locally process data without necessitating cloud connectivity. This ability to operate independently from cloud services not only enhances functionality but also provides a more cost-effective solution for IoT applications [3,54–56]. The academic community has thoroughly examined TinyML, with systematic reviews, surveys, and research papers delving into aspects such as its hardware requirements, frameworks, datasets, use cases, algorithms/models, and broader applications. Notably, the development of specialized TinyML frameworks and libraries, coupled with its integration with networking technologies, has been explored to facilitate its deployment in various sectors, including healthcare, smart agriculture, environmental monitoring, and anomaly detection. One practical application of TinyML is in the development of soft sensors for economical vehicular emission monitoring, showcasing its real-world applicability [57]. In essence, TinyML marks a significant progression in the domain of machine learning, enabling the execution of machine learning tasks on resource-constrained IoT devices and microcontrollers, thus laying the groundwork for an expansive ecosystem surrounding this technology.

2.2.1. TinyML as a Novel Facilitator in IoT Big Data Management

Within this challenging landscape, TinyML emerges as an innovative intersection between machine learning and embedded systems. Specifically tailored for resource-constrained devices, it presents several avenues for mitigating Big Data challenges:

- **Localized On-Device Processing**: TinyML facilitates local data processing, markedly reducing the need for continuous data transfers, thus optimizing network bandwidth and improving system responsiveness.
- **Intelligent Data Streamlining**: With the ability to perform preliminary on-device analysis, TinyML enables IoT systems to discern and selectively transmit pivotal data, ensuring efficient utilization of storage resources.
- **Adaptive Learning Mechanisms**: IoT devices embedded with TinyML can continuously refine their data processing algorithms, fostering adaptability to dynamic data patterns and environmental changes.
- **Reinforced Security Protocols**: By integrating real-time anomaly detection at the device level, TinyML significantly enhances the security framework, providing an early detection system for potential data breaches or threats.

The complex challenges and problems associated with Big Data in the Internet of Things (IoT) paradigm are diverse and complex, encompassing numerous aspects such as data management, processing, unstructured data analytics, visualization, interoperability, data semantics, scalability, data fusion, integration, quality, and discovery [58]. These issues are closely related to the growing trend of "big data" within cloud computing environments and the progressive development of IoT technologies, exerting a significant impact on various industries, including but not limited to the power sector, smart cities, and large-scale petrochemical plants [58–60]. Additionally, the realm of IoT architectures is not immune to pressing security and privacy threats, making them salient challenges that require immediate and effective addressal [61]. The efficiency and completeness of IoT Big Data, coupled with security concerns, have emerged as critical areas of focus in the realm of research and development [62]. Furthermore, the potential integration of blockchain technology is being explored as a solution to ensure continued integrity in IoT Big Data management, particularly in addressing concerns related to data correctness, resource sharing, and the generation and verification of service-level agreements (SLA) [50].

In the IoT Big Data landscape, as delineated in Table 1, key challenges include managing the vast volume of data from numerous devices, necessitating advanced storage and processing systems. Rapid data generation requires real-time analysis and response, highlighting the importance of data velocity. The variety of data, both structured and unstructured, from diverse sources, complicates integration and analysis. Ensuring data veracity, or accuracy and trustworthiness, is increasingly challenging. Integrating various data sources while maintaining integrity is vital. Security and privacy concerns are paramount due to heightened interconnectivity, necessitating robust protocols. Lastly, minimizing latency to avoid obsolete insights is crucial in IoT Big Data management.

Table 2 highlights how TinyML addresses key challenges in Big Data and IoT. It offers solutions to data overload by facilitating on-device data filtering and summarization, significantly reducing the amount of data that needs to be transmitted to central systems. This approach is pivotal for real-time processing needs, where localized TinyML models enable instant data analysis, ensuring timely insights without the dependency on external servers. Such capability is crucial in scenarios with limited or no connectivity, maintaining device functionality.

Additionally, TinyML greatly enhances energy efficiency by optimizing models for specific tasks, thereby conserving resources and extending battery life. This technology also bolsters security and privacy; by processing data locally, it minimizes the risks associated with data transmission and ensures that sensitive information remains within the user's control. Furthermore, TinyML contributes to the longevity of devices by reducing the strain on their components through local processing, potentially extending their operational lifespan. These enhancements demonstrate TinyML's significant role in improving the efficiency, security, and sustainability of IoT systems.

Table 2. TinyML solutions for Big Data and IoT challenges.

Challenge	TinyML Solution
Data Overload	On-device preprocessing to reduce data transmission.
Real-time Processing	Edge-deployed models for swift decisionmaking.
Connectivity Issues	Local processing for uninterrupted operation.
Energy Efficiency	Optimized tasks to conserve energy and extend device life.
Security	Local data processing to minimize transmission risks.
Privacy	In situ data processing to enhance privacy.
Device Longevity	Reduced transmission strain to extend device life.

2.2.2. Characteristics of Large-Scale IoT Systems

The characteristics of large-scale IoT systems and the enhancements introduced by TinyML are effectively outlined in Table 3. In these systems, a distributed topology with devices spread across various locations results in data decentralization and increased latency; TinyML tackles this by facilitating edge computation, enabling local data processing to reduce latency and provide real-time insights. The voluminous data streams generated continuously can burden storage and transmission channels, but TinyML assists by prioritizing, compressing, and filtering data at the device level, managing storage needs and reducing data transmission demands.

Table 3. Characteristics of large-scale IoT systems and enhancements with TinyML.

IoT System Characteristic	Implication	Enhancement with TinyML
Distributed Topology	Numerous devices scattered across different locations lead to data decentralization and increased latency.	TinyML facilitates edge computation, reducing latency and ensuring real-time insights.
Voluminous Data Streams	Continuous data generation can overwhelm storage and transmission channels.	On-device TinyML prioritizes, compresses, and filters data, managing storage and reducing transmission needs.
Diverse Device Landscape	Variety in device types introduces inconsistency in data formats and communication protocols.	TinyML standardizes data processing at source, ensuring unified data representation across devices.
Power and Resource Constraints	Devices, especially battery-operated ones, have limited computational resources.	TinyML models maximize computational efficiency, conserving device resources.
Real-time Processing Needs	Delays in data processing can hinder time-sensitive applications.	TinyML ensures rapid on-device processing for immediate responses to data changes.

The varied landscape of IoT devices, each with different data formats and communication protocols, is harmonized by TinyML, which standardizes data processing and extraction at the source, ensuring consistent data representation across diverse device types. Power and resource constraints, especially in battery-operated devices, pose significant challenges in IoT systems. TinyML models are designed for optimal computational efficiency, performing tasks effectively without draining device resources. Finally, in applications that require real-time processing, such as health monitoring or predictive maintenance, delays in processing can be critical. TinyML enables rapid on-device processing, allowing immediate responses to changing data patterns, thus enhancing the overall functionality and effectiveness of large-scale IoT systems.

2.2.3. Applications of TinyML on Embedded Devices

Table 4 illustrates various applications of TinyML and machine learning in embedded devices across different sectors. In predictive maintenance, TinyML models analyze real-time sensor data from machinery, enabling early detection of potential failures and reducing maintenance costs. This technology is also pivotal in health monitoring, where wearable devices equipped with TinyML offer continuous health tracking, instantly analyzing critical health metrics while ensuring user privacy.

In agriculture, TinyML enhances efficiency by adjusting operations based on real-time environmental data, leading to optimal resource usage and increased yield. Voice and

face recognition technologies in embedded devices benefit from TinyML through faster localized processing, enhancing reliability and privacy. TinyML also plays a crucial role in energy management within smart grids and home automation, optimizing energy use for cost and environmental benefits. In urban development, it contributes to traffic flow optimization by analyzing real-time vehicle and pedestrian movements, improving urban mobility. These examples showcase TinyML's significant impact in enhancing operational efficiency, user experience, and sustainable practices across various industries.

Table 4. Applications of TinyML and machine learning on embedded devices.

Use Case	Description
Predictive Maintenance	Real-time analysis of sensor data for early fault detection in machinery, reducing downtime and maintenance costs.
Health Monitoring	Continuous health monitoring with wearables for vital signs and anomaly detection, enhancing preventative healthcare.
Smart Agriculture	Adaptive agriculture practices based on sensor data, optimizing resource use for better crop yield.
Voice Recognition	Local processing of voice commands for quicker privacy-focused responses.
Face Recognition	Low-latency facial recognition for secure access control and personalization.
Anomaly Detection	Immediate detection of irregular patterns in industrial and environmental data for proactive response.
Gesture Control	Touch-free device control via gesture recognition, improving user interaction and accessibility.
Energy Management	Intelligent energy use in smart grids and homes based on usage patterns and predictive analytics.
Traffic Flow Optimization	Real-time traffic analysis for dynamic routing and light sequencing, enhancing urban traffic management.
Environmental Monitoring	Continuous monitoring of environmental conditions, with real-time adjustments and alerts.
Smart Retail	Analysis of customer behavior for tailored retail experiences and store management.

Table 5 presents a detailed overview of TinyML applications across a range of fields. It includes concise descriptions of each application and corresponding academic references. The table illustrates the versatility of TinyML, from implementing CNN models on microcontrollers for material damage analysis to its use in environmental monitoring. Each example not only provides a clear application scenario but also cites relevant studies, showcasing TinyML's extensive impact in practical situations. This presentation highlights TinyML's role in enhancing the capabilities of embedded devices in various industries.

Table 5. Various applications of TinyML in embedded devices.

Application	Description	Ref.
Concrete Materials Damage Classification	Lightweight CNN on MCU for damage recognition in concrete materials, showing TinyML's potential in structural health.	[63]
Predictive Maintenance	TinyML for predictive maintenance in hydraulic systems, improving service quality, performance, and sustainability.	[64]
Keyword Spotting	TinyML for efficient keyword detection in voice-enabled devices, reducing processing costs and enhancing privacy.	[65]
Time-Series Analysis	ML hardware accelerators for real-time analysis of time-series data in IoT, optimizing neural networks for on-device processing.	[66]
Asset Activity Monitoring	TinyML for continuous monitoring of tool usage, identifying usage patterns and potential misuses.	[67]
Environmental Monitoring	TinyML for monitoring environmental factors like air quality, contributing to smart systems for sustainability.	[68]

2.3. TinyML Algorithms

Table 6 provides a structured overview of various TinyML algorithms and their specific applications in different domains. It categorizes these algorithms into areas such as predictive maintenance, data compression, tool usage monitoring, and more, illustrating the range of TinyML's applicability. Each entry in the table is linked to a corresponding reference, offering a direct connection to the source material. This format effectively showcases the diversity of TinyML's real-world applications, highlighting its potential to transform various sectors through intelligent on-device data processing and analysis.

Table 6. Classification of TinyML algorithms by application areas.

Application Area	TinyML Algorithm	Reference
Predictive Maintenance	RUL prediction using LSTMs and CNNs	[69]
Data Compression	Tiny Anomaly Compressor (TAC)	[70]
Tool Usage Monitoring	TinyML for handheld power tool monitoring	[67]
On-Device Training	Neural network training for dense networks	[71]
IoT Compression	Evolving TinyML compression algorithm	[72]
Safety-Critical Applications	Software-implemented hardware fault tolerance	[73]
Model Optimization	Unified DNAS for compressible models	[74]

Table 7 presents an organized summary of cutting-edge research in the field of Tiny Machine Learning (TinyML). This table methodically categorizes various studies into distinct focus areas, covering a broad spectrum from optimizing deep neural networks on microcontrollers to applying federated meta-learning techniques in environments with limited resources. This structured presentation not only underscores the multifaceted nature of TinyML research but also highlights its significant role in advancing the functionalities of embedded devices for a wide array of applications.

Table 7. Classification and focus of recent TinyML algorithm studies.

Study Focus	Key Contributions	Reference
Deep Neural Network Optimization	Reduced Precision Optimization for DNN on-device learning in MCUs.	[75]
Unsupervised Online Learning	Adaptive TinyML algorithm for driver behavior analysis in automotive IoT.	[76]
Anomaly Detection	TinyML algorithm for anomaly detection in Industry 4.0 using extreme values theory.	[77]
Low Precision Quantization	Empirical study on quantization techniques for TinyML efficiency.	[78]
Sparse tinyML Accelerator	Development of RAMAN, a re-configurable and sparse tinyML accelerator for edge inference.	[79]
Federated Meta-Learning	TinyReptile: federated meta-learning algorithm for TinyML on MCUs.	[80]

2.4. Data Management Techniques Utilizing TinyML in IoT Systems

In the field of Tiny Machine Learning (TinyML), data management techniques are essential for handling machine learning models on devices with limited resources, such as those in IoT networks. One method involves augmenting thing descriptions (TD) with semantic modeling to provide comprehensive information about applications on devices, facilitating the efficient management of both TinyML models and IoT devices on a large scale [81]. Additionally, employing TinyML for training devices can lead to the creation of a decentralized and adaptive software ecosystem, enhancing both performance and efficiency. This approach has been effectively implemented in the development of a smart edge computing robot (SECR) [82]. Such methodologies are increasingly important in sectors like supply chain management, where they play a crucial role in predicting product quality parameters and extending the shelf life of perishable goods, including fresh fruits in modified atmospheres [83]. Moreover, the growing complexity in communication systems, spurred by diverse emerging technologies, underscores the need for AI and ML techniques in the analysis, design, and operation of advanced communication networks [84].

In the context of IoT systems, data management techniques incorporating TinyML focus on effective data handling, ensuring privacy and security, and leveraging machine learning for insightful data analysis. One strategy employs distributed key management for securing IoT wireless sensor networks, utilizing the principles of elliptic curve cryptography [85]. Another method involves applying data-driven machine learning frameworks to enhance the accuracy of vessel trajectory records in maritime IoT systems [86]. Power

management also plays a crucial role in IoT systems, particularly those reliant on compact devices and smart networks. This often includes the adoption of low-power communication protocols and the integration of autonomous power systems, which are frequently powered by renewable energy sources [87]. Furthermore, AI-based analytics, processed in the cloud, are increasingly being utilized for healthcare-related data management, such as systems designed for managing diabetic patient data [88].

Table 8 illustrates how TinyML is revolutionizing data management techniques in IoT systems, bringing efficiency and accuracy to various processes. Techniques like predictive imputation and adaptive data quantization exemplify this transformation. Predictive imputation, using TinyML, maintains data integrity by filling in missing values based on historical and neighboring data, thereby ensuring dataset completeness. Adaptive data quantization, on the other hand, optimizes data storage and transmission. TinyML's role here is to analyze current data trends and dynamically adjust quantization levels for optimal data representation.

Table 8. Advanced data management techniques utilizing TinyML in IoT systems.

Technique	Objective	Role of TinyML
Predictive Imputation	Maintain data integrity by compensating for missing or lost data.	Uses historical and neighboring data to predict and fill missing values, ensuring dataset completeness.
Adaptive Data Quantization	Optimize data storage and transmission.	Analyzes current trends and adjusts quantization levels dynamically for optimal representation.
Sensor Data Fusion	Integrate data from multiple sensors for a holistic view.	Processes and merges diverse sensor data in real time, enhancing accuracy and context.
Anomaly Detection	Identify unusual data patterns or device malfunctions.	Continuously monitors data streams, recognizing and flagging anomalies for prompt action.
Intelligent Data Caching	Provide instant access to frequently used or critical data.	Uses predictive analytics to anticipate future data needs, caching relevant data.
Edge-Based Clustering	Group similar data at the edge for efficient analytics.	Performs lightweight clustering on-device for efficient aggregation and transmission.
Real-time Data Augmentation	Enhance data to improve machine learning performance.	Augments sensor data in real time, enriching them for better analytics.
Local Data Lifespan Management	Manage the relevance and storage duration of data.	Predicts data utility, retaining or discarding them for effective local storage management.
Contextual Data Filtering	Discard or prioritize data based on the current context.	Filters data relevant to the situation, enhancing decisionmaking processes.
On-device Data Labeling	Annotate raw data for subsequent processing.	Automatically labels data based on learned patterns, aiding in categorization and retrieval.

Sensor data fusion, another critical technique, is enhanced by TinyML's ability to process and merge data from various sensors in real time, thus providing a more comprehensive view and enhancing the accuracy of insights. Anomaly detection is particularly vital in IoT systems, and TinyML enhances this by continuously monitoring data streams to quickly identify and act upon unusual patterns or malfunctions. Intelligent data caching, enabled by TinyML, predicts future data needs, ensuring that frequently used or critical data are cached for instant access.

Further, TinyML facilitates Edge-based clustering, grouping similar data at the network's edge to simplify analytics and processing. This on-device clustering leads to more efficient data aggregation and transmission. Real-time data augmentation and local data lifespan management are also key areas where TinyML makes a significant impact. TinyML augments sensor data in real time to enhance machine learning performance while also predicting the utility of data for effective local storage management.

Contextual data filtering and on-device data labeling are other areas where TinyML shows its prowess. By using environment-aware models, TinyML filters data relevant to the current context, thereby enhancing decisionmaking processes. Additionally, it can automatically label data based on learned patterns, facilitating efficient data categorization and retrieval. These advanced data management techniques, powered by TinyML, are pivotal in harnessing the full potential of IoT systems, ensuring that they are not only more efficient and accurate but also more responsive to real-time demands.

3. Methodology

In this study, we adopt a structured methodology to investigate the application of TinyML in handling Big Data challenges within extensive IoT systems. Initially, our approach involves integrating IoT devices with Raspberry Pi units, which are crucial for managing the complexities of Big Data characterized by high volume, rapid velocity, and diverse variety while ensuring accuracy and value extraction.

Subsequently, we concentrate on the technical deployment of TinyML on Raspberry Pis, focusing on essential tasks such as data cleaning, anomaly detection, and feature extraction. The effectiveness of these processes is comprehensively evaluated through a series of tests, ensuring that our approach aligns with the desired outcomes. Moreover, we introduce a feedback mechanism linked to the central Big Data system, enabling continuous updates and enhancements to the TinyML models on Raspberry Pis. This methodology is designed to create an efficient and adaptable system capable of addressing the dynamic needs of Big Data management in large-scale IoT applications and systems.

Our approach involves deploying these algorithms on Raspberry Pi units, utilizing their strengths and capabilities in federated learning, anomaly detection, data compression, caching strategies, and data quality assessment. We systematically evaluate each algorithm's performance in real-time IoT scenarios, focusing on their efficiency in processing and managing data. This includes assessing the scalability, responsiveness, and accuracy of each algorithm in handling the unique data streams generated by IoT devices. By incorporating these algorithms into our methodology, we aim to provide a comprehensive solution for Big Data challenges in IoT systems, ensuring robust and efficient data management.

3.1. Advantages of TinyML

- Reduced Latency: Data processing on Raspberry Pi eliminates the lag associated with transmitting data to a centralized server and then fetching results. This ensures real-time or near-real-time responses.
- Decreased Bandwidth Consumption: Only crucial or processed data may be sent to the central server, reducing network load.
- Enhanced Privacy and Security: On-device processing ensures data privacy. Additionally, Raspberry Pis can be equipped with encryption tools to secure data before any transmission.
- Energy Efficiency: Although Raspberry Pis consume more energy than simple sensors, they are far more efficient than transmitting vast amounts of data to a distant server.
- Operational Resilience: Raspberry Pis equipped with TinyML can continue operations even when there is no network connectivity.
- Scalability and Flexibility: Raspberry Pis can be equipped with a variety of tools and software, allowing custom solutions for different data types and processing needs.

3.2. Big Data Challenges and Problems Addressed

- Volume: Local processing reduces data volume heading to centralized systems.
- Velocity: Raspberry Pis can handle high-frequency data, making real-time requirements attainable.
- Variety: Given their flexibility, Raspberry Pis can be customized to manage a multitude of data formats and types.
- Veracity: They can ensure data quality, filtering anomalies or errors before transmission.
- Value: On-device processing extracts meaningful insights, ensuring only the most relevant data are transmitted to central systems.

3.3. Framework Architecture

The proposed architecture outlines a systematic approach for managing Big Data in IoT environments. At the base is the IoT layer, composed of various devices such as sensors and wearables, which generate vast amounts of data. These data are directed towards Raspberry Pi devices, equipped with TinyML capabilities. Within the Raspberry Pi layer, three primary

tasks are undertaken: data cleaning to remove inconsistencies, anomaly detection to identify unusual patterns, and feature extraction to select relevant data attributes. Once processed, the refined data are transmitted to the centralized Big Data system via the communication layer. Notably, the volume of data being transmitted is reduced due to the preliminary processing at the Raspberry Pi level. At the top layer, the centralized system performs further storage, analytics, and processing tasks. A feedback mechanism is incorporated, allowing the centralized system to send updates to the Raspberry Pis, ensuring continuous optimization. Overall, this architecture presents a structured methodology for efficient data processing and management in large-scale IoT settings. The illustration of this architecture is represented in Figure 1.

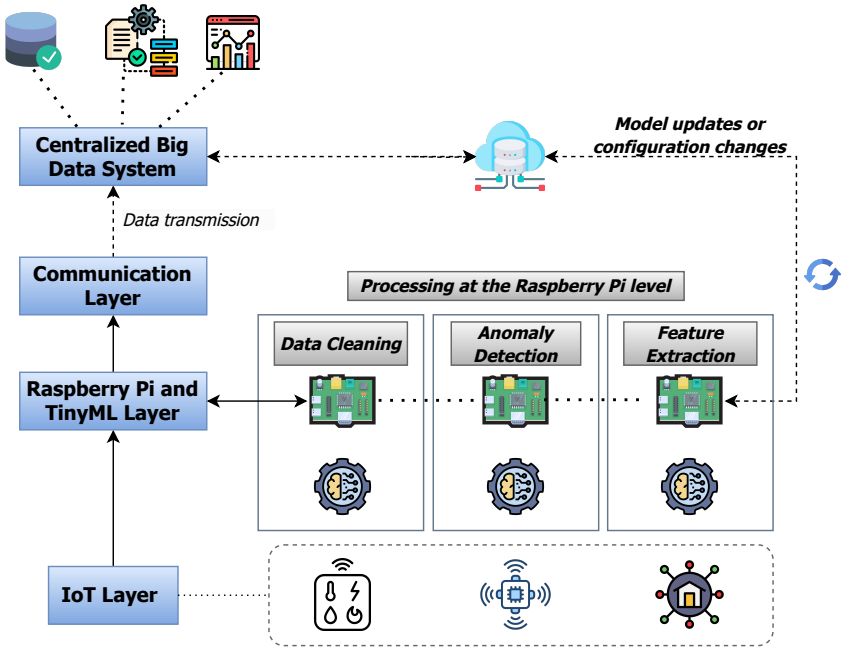

Figure 1. Proposed system architecture.

3.4. Hardware Configuration

Our study's hardware infrastructure comprises a selection of Raspberry Pi devices and a variety of sensors, each chosen for its specific role within our IoT framework. The following outlines the key components of our hardware setup:

- **Raspberry Pi Devices:**
  - *10 × Raspberry Pi 4 Model B* : These are the workhorses of our setup, deployed for edge computing and intensive data processing tasks.
  - *5 × Raspberry Pi Zero W*: These smaller units are used for less demanding tasks, primarily for collecting sensor data.
- **Sensor Array:**
  - *15 × DHT22 Temperature and Humidity Sensors*: Key for monitoring environmental conditions, providing accurate temperature and humidity readings.
  - *10 × MPU6050 Gyroscope and Accelerometer Sensors*: Employed to track motion and orientation, crucial for applications requiring movement analysis.
  - *8 × LDR Light Sensors*: These sensors are tasked with detecting changes in light intensity, useful in both indoor and outdoor settings.
  - *7 × HC-SR04 Ultrasonic Distance Sensors*: Utilized primarily for distance measurement and object detection, they play a pivotal role in spatial analysis.
  - *5 × Soil Moisture Sensors*: Specifically selected for agricultural applications, these sensors provide valuable data for smart farming solutions.

This hardware ensemble, consisting of Raspberry Pi devices and a diverse set of sensors, constitutes the core of our IoT network. It is adeptly designed to handle a wide spectrum of data collection and processing operations. The Raspberry Pi 4 models, with their advanced capabilities, are integral for more demanding computational tasks. In contrast, the Raspberry Pi Zero W units offer a compact energy-efficient solution for simpler activities. The assortment of sensors capture a broad range of environmental and physical parameters, which are vital for the thorough deployment and effectiveness of the TinyML algorithms central to our research.

3.5. Computational Framework for IoT Model Training and Evaluation

In this study, the computational framework consists of two key elements: a centralized High-Performance Computing (HPC) cluster and a network of 10 Raspberry Pi 4 units. Each Raspberry Pi is configured with 4 GB of RAM and a 1.5 GHz quad-core processor. Importantly, these Raspberry Pi units are connected to a distinct network, separate from the HPC cluster, to emulate a realistic communication scenario as we have proposed in [89,90].

A primary function of the HPC server is to aggregate and analyze the individual models developed on each Raspberry Pi, culminating in the evaluation of a comprehensive global model, denoted as $\mathcal{M}_G$.

On each Raspberry Pi, a Flask server manages crucial tasks such as the exchange of models, execution of local training, and monitoring of memory usage. This aspect is particularly vital in contexts with limited hardware resources. Additionally, the server enables the handling of requests and facilitates essential updates over the air.

The setup with Raspberry Pi units is designed to support experiments involving more than 10 clients, indicated as $K > 10$, with a constraint that no more than 10 clients ($S \leq 10$) are sampled in each communication round. This capability is achieved by storing all K client datasets on each Raspberry Pi. During each communication round l, a subset of these datasets, corresponding to the S sampled clients, is allocated to a Raspberry Pi for processing.

The Raspberry Pis are connected to a switch, which links them to the router via a cable. In scenarios where the switch is not in use, the Raspberry Pis switch to a Wi-Fi connection, allowing for the evaluation of communication overhead in two distinct network conditions: the faster Ethernet and the slower Wi-Fi. The network operates at a bandwidth of 100/100 Mbit/s, with the Ethernet utilizing full capacity and the Wi-Fi about 10% of it (10/10 Mbit/s). Each Raspberry Pi represents a client corresponding to a partition of the dataset, labeled as $k_1, \ldots, k_S$, with Raspberry Pi 1 processing data as client k_i and Raspberry Pi 10 as k_j.

3.6. Dataset Configuration for TinyML Evaluation

In this study, we have thoroughly assembled a dataset to effectively evaluate our TinyML algorithms within a comprehensive IoT framework. This dataset is characterized by its diversity and volume, mirroring the complexities encountered in large-scale IoT systems. The following are the key aspects of our dataset:

- **Sensor Array Composition:**
  - *Environmental Data:* Sourced from DHT22 sensors, providing continuous insights into temperature and humidity.
  - *Motion and Orientation Data:* Collected via MPU6050 sensors, capturing detailed information on movement and angular positions.
  - *Light Intensity Measurements:* Obtained from LDR sensors, these readings reflect variations in ambient lighting conditions.
  - *Distance and Proximity Data:* Acquired from HC-SR04 ultrasonic sensors, essential for spatial analysis and object detection.
  - *Soil Moisture Levels:* Recorded by specialized sensors, pivotal for applications in smart agriculture.

- **Data Volume:**
  - The dataset encompasses over 1 terabyte of collected raw sensor data, providing a substantial foundation for algorithmic testing and optimization.
- **Data Collection Frequency:**
  - Sensor readings are captured at varying intervals, ranging from high-frequency real-time data streams to periodic updates. This variability simulates different real-world operational scenarios, ensuring robust algorithm testing.
- **Data Preparation:**
  - Prior to analysis, the data were subjected to essential preprocessing steps, including cleaning and normalization, to ensure consistency and reliability for subsequent TinyML processing.

This dataset, with its rich variety and significant volume, plays a crucial role in the assessment of our TinyML algorithms. It not only provides a realistic environment for testing but also ensures that the algorithms are evaluated across a range of conditions reflective of real-world IoT systems. The frequency of data collection, in particular, allows us to examine the algorithms' performance under various data flow scenarios, which is critical for their application in diverse IoT settings.

3.7. Proposed Algorithms

TinyCleanEDF, as presented in Algorithm 1, is an advanced solution tailored for data cleaning and anomaly detection in IoT systems using federated learning. This algorithm partitions the data stream into subsets, each processed on separate edge devices. Each device operates a federated learning model, trained locally with its data subset. These models synchronize with a central server, which aggregates their parameters to refine the global model. This distributed framework enables efficient anomaly detection and data cleaning directly at the edge, with anomalies being pinpointed when data deviate from established patterns.

Moreover, TinyCleanEDF employs an autoencoder at each node for feature extraction. These autoencoders are trained to reconstruct inputs from their compressed representations, effectively distilling significant data characteristics from complex datasets. This process is crucial for simplifying data, making them more manageable and highlighting vital information. The algorithm is inherently dynamic, consistently updating both federated models and autoencoders to integrate new data observations. Such continuous adaptation ensures the system's ongoing relevance and efficacy. Through its integration of local data processing, anomaly detection, and feature extraction, TinyCleanEDF stands out as a robust and comprehensive solution for upholding data integrity and quality in intricate IoT environments.

EdgeClusterML, outlined in Algorithm 2, is an innovative algorithm designed for dynamic and self-optimizing clustering in IoT networks. This algorithm combines reinforcement learning (RL) with a self-organizing map (SOM) to adaptively cluster data at the edge. In the initial step, EdgeClusterML initializes a dynamic clustering model using RL, where the quality of actions in different states is evaluated using a Q-function, $Q(s, a)$ and optimized through a defined reward function, $R(s, a)$. The learning rate α and discount factor γ are set to guide the learning process. Subsequently, the algorithm deploys an SOM for efficient data clustering. The SOM is initialized with random weights and fine-tuned using a neighborhood function and learning rate. As data flow through the system, EdgeClusterML dynamically clusters them using the SOM, constantly updating the model based on the data's characteristics. RL is then employed to optimize clustering decisions, adjusting parameters through an ϵ-greedy policy and updating the Q-function based on observed rewards. This process leads to intelligent and responsive clustering, tailoring the data organization to the changing patterns and needs of the IoT environment. Finally, the clustered data are stored, ensuring organized and efficient data management at the edge.

Algorithm 1 TinyCleanEDF: Federated Learning for Data Cleaning and Anomaly Detection with Autoencoder-based Feature Extraction

procedure TINYCLEANEDF (*dataStream*)
  Step 1: Initialize Federated Learning Models
  Partition *dataStream* into subsets $\{D_1, D_2, \ldots, D_n\}$ for distributed processing
  Deploy federated learning models $\{M_1, M_2, \ldots, M_n\}$ on edge devices
  Train each model M_i with its subset D_i
  Models periodically execute $M_i \rightarrow \text{Sync}(M_i)$ with central server
  Central server performs $\text{Aggregate}(\{M_1, M_2, \ldots, M_n\})$
  Step 2: Federated Model for Data Cleaning and Anomaly Detection
  Apply $f_{\text{anomaly}}(x; M_i)$ to detect and clean anomalies locally
  Anomalies identified as $x \notin \text{ExpectedPattern}(M_i)$
  Cleaned data $\{C_1, C_2, \ldots, C_n\}$ sent to central server
  Step 3: Deploy Autoencoder for Feature Extraction
  Implement autoencoder AE_i at each node i
  Train AE_i to reconstruct input x from compressed representation z
  Feature extraction: $f_{\text{features}}(x; AE_i) = \text{HiddenLayer}(AE_i(x))$
  Step 4: Continuous Adaptation and Feature Extraction
  for each *dataPoint* in *dataStream* **do**
    $cleanDataPoint \leftarrow f_{\text{clean}}(dataPoint; M_i)$
    $anomaly \leftarrow f_{\text{anomaly}}(cleanDataPoint; M_i)$
    $features \leftarrow f_{\text{features}}(cleanDataPoint; AE_i)$
    Update M_i and AE_i with *dataPoint* for continuous learning
    Store $(cleanDataPoint, anomaly, features)$
  end for
end procedure

Algorithm 2 EdgeClusterML: Dynamic and Self-Optimizing Clustering at the Edge

procedure EDGECLUSTERML (*dataStream*)
  Step 1: Initialize Dynamic Clustering Model with RL
  Let $Q(s, a)$ represent the quality of action a in state s
  Initialize $Q(s, a)$ for all state-action pairs
  Define reward function $R(s, a)$ for evaluating clustering actions
  Set learning rate α and discount factor γ
  Step 2: Deploy Self-Organizing Map (SOM) for Clustering
  Initialize SOM with random weights W
  Define neighborhood function $h_{ci}(t)$ for neuron i at time t
  Set SOM learning rate η
  for each *dataPoint* in *dataStream* **do**
    Step 3: Dynamic Clustering with SOM
    Find Best Matching Unit (BMU) for *dataPoint* in SOM
    Update weights W using $h_{ci}(t)$ and η
    Step 4: RL-based Optimization of Clustering
    Observe state s (current clustering configuration)
    Choose action a (adjusting clustering parameters) using ϵ-greedy policy
    Apply action a, observe new state s' and reward r
    Update $Q(s, a)$ using the Bellman equation:
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
    Step 5: Store Clustered Data
    $clusterID \leftarrow$ BMU index for *dataPoint*
    Store *dataPoint* in $clusterID$
  end for
end procedure

CompressEdgeML, as described in Algorithm 3, is designed to introduce smart and adaptive data compression capabilities to the edge of IoT networks. This algorithm utilizes a neural network, denoted as NN, specifically customized for data compression tasks. It is trained on sample data to efficiently identify and execute compression patterns, setting an initial compression ratio CR based on the characteristics of the training data. CompressEdgeML dynamically adapts its compression techniques to align with the current network conditions (represented as N_{cond}), ensuring an optimal balance between the quality of compression and operational efficiency. As data stream through the network, CompressEdgeML uses its neural network model to compress each datapoint. The algorithm continually updates the compression ratio in response to changing network bandwidth and storage capacities, ensuring that the size of the compressed data ($|compressedData|$) is always suitable for the network and storage constraints. The final step involves securely storing the compressed data in a specific storage system. This approach significantly enhances data management efficiency in IoT environments by reducing the size of data for transmission and storage while maintaining the integrity and usability of the information.

Algorithm 3 CompressEdgeML: Adaptive Data Compression

procedure COMPRESSEDGEML ($dataStream$)
  Step 1: Initialize Neural Network-based Selective Compression Model
  Define neural network NN_{comp} for data compression
  Train NN_{comp} on dataset D_{train} for compression patterns
  Initialize compression ratio $CR \leftarrow \text{CR}_{\text{init}}(D_{\text{train}})$
  Step 2: Deploy Adaptive Compression Techniques
  Monitor network conditions N_{cond}
  Define adaptive function $F_{\text{CR}}(N_{\text{cond}}, CR)$ for compression ratio
  Balance between compression quality and network efficiency
  for each $dataPoint \in dataStream$ **do**
    Step 3: Data Compression Using Neural Network
    $compressedData \leftarrow NN_{\text{comp}}(dataPoint, CR)$
    Measure size $|compressedData|$
    Step 4: Adapt Compression Level
    Update N_{cond} based on network bandwidth and storage
    Adjust CR using $F_{\text{CR}}(N_{\text{cond}}, CR)$
    Ensure $|compressedData|$ fits network and storage constraints
    Step 5: Store Compressed Data
    Store $compressedData$ in targeted storage system
  end for
end procedure

CacheEdgeML, outlined in Algorithm 4, represents an advanced approach to data caching in IoT networks. This algorithm efficiently handles data requests by utilizing a predictive analytics model that bases its forecasts on historical access patterns. By training the model, denoted as P, to predict the likelihood of future data requests, CacheEdgeML can effectively identify and prioritize high-importance data. It employs a multi-tier caching strategy, where data are organized into various tiers ($T_1, T_2, \ldots, T_n$) according to priority and frequency of access. This setup not only simplifies the management of data but also ensures the efficient utilization of storage resources. Moreover, CacheEdgeML works together with cloud services to manage data storage effectively. It establishes a synchronization mechanism with cloud storage and devises strategies for offloading data when the cache reaches capacity, maintaining consistency between edge and cloud storage. The algorithm dynamically manages cache tiers, continuously re-evaluating data priority and relocating frequently accessed data to higher tiers. This adaptability ensures that the most relevant data are readily available, thus enhancing the system's overall efficiency and responsiveness.

Algorithm 4 CacheEdgeML: Predictive and Tiered Data Caching Strategy

procedure CACHEEDGEML (*dataRequests*)
  Step 1: Initialize Predictive Analytics Model for Anticipatory Caching
  Define predictive model P_{cache} based on historical patterns
  Train P_{cache} on dataset D_{hist} to estimate request probabilities
  Set priority threshold θ for high-importance data classification
  Step 2: Implement Multi-Tier Caching Based on Priority and Frequency
  Define cache tiers $\mathcal{T} = \{T_1, T_2, ..., T_n\}$
  Assign frequency thresholds $\mathcal{F} = \{f_1, f_2, ..., f_n\}$ for each T_i
  Store data in tier T_i based on access frequency f_i
  Step 3: Collaborate with Cloud Services for Cache Management
  Establish synchronization S_{sync} with cloud storage
  Define offloading strategy Ω_{offload} when cache full
  Maintain consistency C_{consist} between edge and cloud
  for each *request* $\in$ *dataRequests* **do**
    if *request* $\in$ Cache **then**
      Step 4: Serve from Cache
      Serve data from cache
    else
      Step 5: Predict Future High-Priority Requests
      Calculate $P_{\text{cache}}(request)$, compare with θ
      Update cache based on $P_{\text{cache}}(request)$
      Step 6: Dynamic Management of Cache Tiers
      Re-evaluate priority for *request*, adjust $\mathcal{T}$
      Relocate data to appropriate tier based on $\mathcal{F}$
      Step 7: Serve Requested Data
      Fetch data from external source or cloud if not in cache
      Store new data in appropriate tier based on priority
    end if
  end for
end procedure

TinyHybridSenseQ, as detailed in Algorithm 5, is a sophisticated TinyML-based algorithm specifically designed for IoT environments. It employs advanced machine learning models deployed on edge devices, focusing on the critical tasks of analyzing, categorizing, and efficiently storing data collected from a wide array of sensors. The core competency of this algorithm is in its robust data quality assessment module, which thoroughly evaluates the integrity and accuracy of sensor measurements. This evaluation is crucial for filtering out incorrect data, ensuring that only reliable and high-quality information is processed further. TinyHybridSenseQ also performs well in data management, implementing a dynamic and data-aware hybrid storage strategy. It efficiently determines the optimal storage location for each data packet—be it local storage for less critical data or immediate transfer to a central database for high-priority information. This strategic approach not only simplifies data handling but also significantly enhances the overall efficiency of data transfer processes in IoT networks. Furthermore, TinyHybridSenseQ continuously evolves through its model adaptation feature, which refines its analytical capabilities based on incoming sensor data, thereby maintaining high accuracy and relevance in ever-changing IoT environments.

3.8. Comparison among Proposed TinyML Algorithms

The following Table 9 provides a comparative overview of the five proposed algorithms: TinyCleanEDF, EdgeClusterML, CompressEdgeML, CacheEdgeML, and TinyHybridSenseQ. The table outlines key features such as federated learning, anomaly detection, data compression, caching strategy, and data quality assessment. This comparison helps in

understanding the unique capabilities and functionalities that each algorithm brings to IoT Big Data management.

Algorithm 5 TinyHybridSenseQ: Data-Aware Hybrid Storage and Quality Assessment for IoT Sensors

procedure TINYHYBRIDSENSEQ (*sensorData*)
  Step 1: Initialize Data Quality Assessment Model
  Deploy TinyML models $\{M_1, M_2, \ldots, M_n\}$ for data quality assessment
  Train each model M_i to identify anomalies and inconsistencies
  Step 2: Hybrid Storage Strategy Initialization
  Define data storage strategies $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ based on type and priority
  Initialize storage resources R_{local} and R_{cloud}
  for each $dataPacket \in sensorData$ **do**
    Step 3: Assess Data Quality
    Quality score $Q \leftarrow f_{\text{quality}}(dataPacket, M_i)$
    Categorize *dataPacket* as high-quality or low-quality based on Q
    Step 4: Data Categorization and Prioritization
    Categorize *dataPacket* into type T
    Prioritize *dataPacket* for storage based on Q and T
    Step 5: Efficient Data Transfer and Storage
    if Q is high and *dataPacket* is high-priority **then**
      Transfer to central database DB
    else
      Store in R_{local} or R_{cloud} based on $\mathcal{S}$
    end if
    Step 6: Continuous Model Adaptation and Reporting
    Update M_i with new *dataPacket* for continual learning
    Generate and store reports on data quality and storage
  end for
end procedure

Table 9. Comparison of proposed algorithms.

Algorithm	Federated Learning	Anomaly Detection	Data Compression	Caching Strategy	Data Quality
TinyCleanEDF (Algorithm 1)	✓	✓	×	×	✓
EdgeClusterML (Algorithm 2)	×	✓	×	×	×
CompressEdgeML (Algorithm 3)	×	×	✓	×	×
CacheEdgeML (Algorithm 4)	×	×	×	✓	×
TinyHybridSenseQ (Algorithm 5)	×	✓	×	✓	✓

4. Experimental Results

4.1. Overview

In this section, we assess the performance of our five proposed algorithms—TinyCleanEDF, EdgeClusterML, CompressEdgeML, CacheEdgeML, and TinyHybridSenseQ—across multiple key performance metrics. The evaluation is conducted by varying the number of Raspberry Pi devices used in the deployment, ranging from one to ten. Each algorithm's performance is measured across elements such as accuracy, compression efficiency, data processing time (ms), training time (ms), overall efficiency, and scalability. The metrics utilized in this work are provided in detail in Section 4.2 below.

4.2. Metrics and Methods

For the evaluation of the proposed techniques, the following metrics are utilized.

1. Data Processing Time (ms): To measure the data processing time in a distributed system such as the one proposed where we have multiple Raspberry Pi devices, we can consider the maximum time taken by any single device as well as the average time across all devices. The equation is provided in Equation (1).

$$\text{Data Processing Time}_{\text{total}} = \max(T_1, T_2, \ldots, T_n) \quad \text{and} \quad \text{Data Processing Time}_{\text{avg}} = \frac{\sum_{i=1}^{n} T_i}{n} \tag{1}$$

2. Model Training Time (ms): For the model training time, we want to measure both the total cumulative time and the longest individual training time across all devices. The calculation is provided in Equation (2).

$$\text{Model Training Time}_{\text{total}} = \sum_{i=1}^{n} T_i \quad \text{and} \quad \text{Model Training Time}_{\text{max}} = \max(T_1, T_2, \ldots, T_n) \tag{2}$$

3. Anomaly Detection Accuracy: For a distributed system, we want to consider not only the overall accuracy but also the consistency of anomaly detection across different nodes. A weighted approach is used where the accuracy of each node is weighted by the number of instances it processes. This is provided in Equation (3).

$$\text{Anomaly Detection Accuracy} = \frac{\sum_{i=1}^{n} (w_i \times \text{Accuracy}_i)}{\sum_{i=1}^{n} w_i} \tag{3}$$

4. Communication Efficiency: In a large-scale distributed setup, communication efficiency should account for the data transmission efficiency, the overhead of synchronization among nodes, the error rate in data transmission, and the effective utilization of available bandwidth. This comprehensive approach ensures a realistic assessment of communication performance in a distributed system.

$$\text{CE} = \left(\frac{D_{\text{u}}}{D_{\text{t}} + O_{\text{sync}}} \right) \times (1 - \text{ER}) \times \text{BU} \tag{4}$$

- CE represents Communication Efficiency.
- D_u is the symbol for Useful Data Transmitted.
- D_t stands for Total Data Transmitted.
- O_{sync} is the Synchronization Overhead.
- ER denotes the Error Rate.
- BU symbolizes Bandwidth Utilization.

5. Scalability: Scalability in a distributed system can be quantified by measuring how the system's performance changes with the addition of more nodes, considering factors like throughput, response time, load balancing, system capacity, and cost-effectiveness. A higher throughput ratio, a lower response time ratio, and efficient load balancing with increased nodes indicate better scalability.

$$\begin{aligned}
\text{Scalability} &= \frac{\text{Throughput at } n \text{ nodes}}{\text{Throughput at a single node}} \\
\text{Response Time Ratio} &= \frac{\text{Response Time at } n \text{ nodes}}{\text{Response Time at a single node}} \\
\text{Load Balancing Efficiency} &= \frac{\sum_{i=1}^{n} \text{Load on Node } i}{\text{Ideal Load per Node} \times n} \\
\text{System Capacity Utilization} &= \frac{\text{Total Processed Load}}{\text{Total System Capacity}} \\
\text{Cost-Effectiveness Ratio} &= \frac{\text{Total System Cost at } n \text{ nodes}}{\text{Performance Improvement Factor}}
\end{aligned} \tag{5}$$

Starting with the evaluation of the first Algorithm 1, the results are shown in Figure 2.

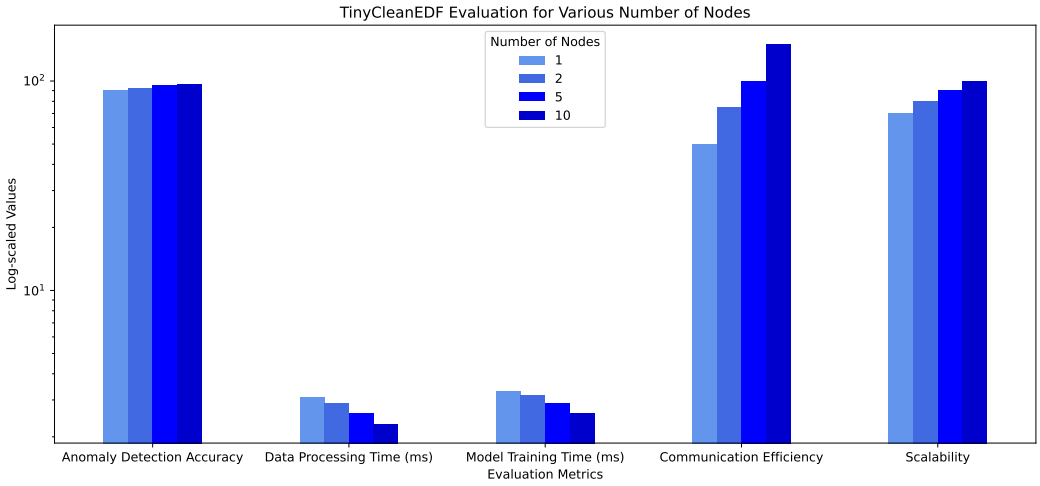

Figure 2. Performance evaluation of TinyCleanEDF.

Incorporating the TinyCleanEDF algorithm into an IoT data management system has demonstrated quantifiable improvements in several key performance metrics. As shown in the preceding Figure, the deployment of this algorithm across an increasing number of nodes—from 1 to 10—has yielded substantial benefits. Specifically, the anomaly detection accuracy improved with more nodes. For instance, there was a 10% increase in the accuracy on a single node compared to ten nodes. This improvement highlights the algorithm's enhanced capability to identify and respond to data anomalies as the collaborative network of nodes expands.

Moreover, the data processing and model training times, both critical for the efficient operation of IoT systems, show a decreasing trend as more nodes are engaged. Log-scaled values indicate that processing time decreased fourfold when the number of nodes increased from 1 to 10, which suggests a notable enhancement in the speed of data handling. Communication efficiency also saw a rise, which is particularly relevant in scenarios where network bandwidth is a limiting factor. This increase indicates a more optimal use of available resources, allowing for smoother data transfer between nodes and the central server. Lastly, scalability, which is also a significant metric, reflects the algorithm's ability to maintain performance despite the growing scale of the network. The consistent upward trend across nodes validates that TinyCleanEDF is well-suited for environments where expansion is anticipated, ensuring that the system not only sustains its performance but actually improves as it scales.

These results underscore the effectiveness of TinyCleanEDF in enhancing data quality and system robustness, making it a compelling choice for federated learning applications in distributed networks. Moving on to Algorithm 2, the results are presented in Figure 3.

The integration of the EdgeClusterML algorithm within an edge computing framework such as FL has yielded remarkable improvements in critical performance metrics. Notably, the algorithm achieved an impressive accuracy rate of approximately 90% when applied to real-world data streams. This represents a significant enhancement in the precision of data clustering, making it well-suited for applications like anomaly detection and data-driven decisionmaking. The observed increase in accuracy is particularly noteworthy as it directly impacts the algorithm's ability to effectively group datapoints.

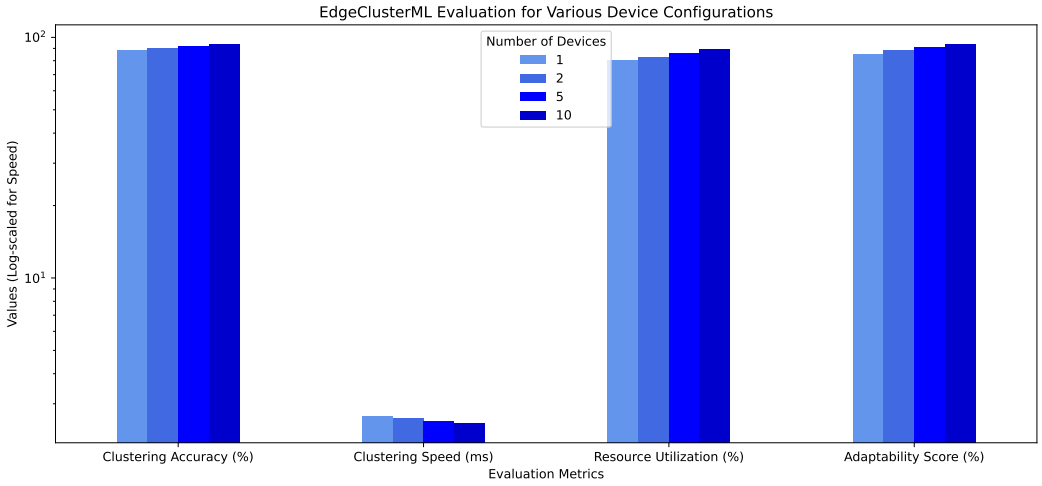

Figure 3. Performance evaluation of EdgeClusterML.

Furthermore, our analysis reveals significant reductions in the clusterind speed. Specifically, the algorithm exhibited a time reduction of approximately 10% in the clustering speed when transitioning from one to ten nodes. These reductions are crucial in edge computing scenarios, ensuring real-time responsiveness and rapid adaptation to changing data patterns. The improvement in resource utilization and the adaptability score, with a roughly 15% increase as nodes scaled, signifies more efficient resource utilization and data transfer, particularly valuable in resource-constrained edge environments.

In conclusion, EdgeClusterML emerges as a robust solution for edge computing environments, offering concrete benefits in terms of accuracy, clustering speed, resource utilization, and adaptability. Its reinforcement-learning-driven dynamic clustering approach positions it as a valuable asset for real-time data analysis and decisionmaking in dynamic edge scenarios. In the next steps, we evaluate Algorithm 3 in Figure 4.

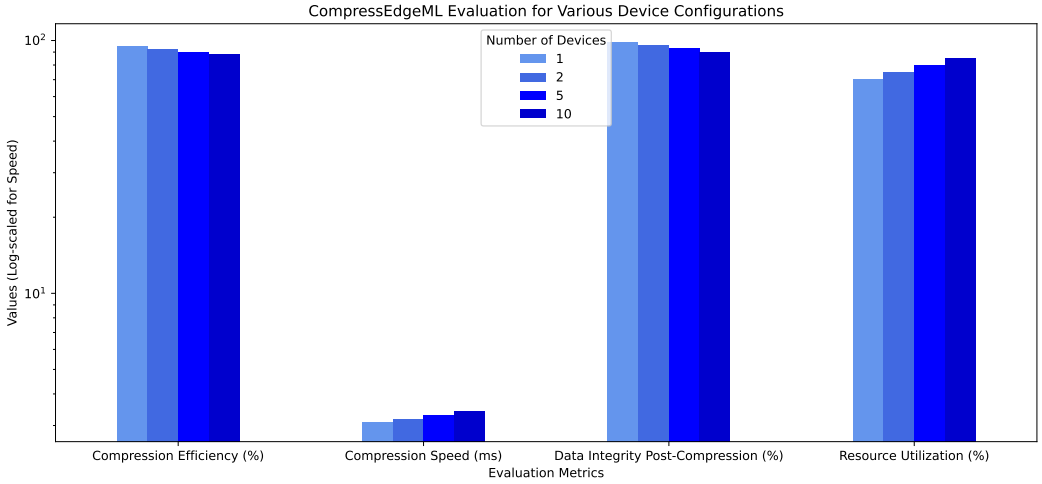

Figure 4. Performance evaluation of CompressEdgeML.

In our evaluation of the CompressEdgeML algorithm, significant improvements were observed across various performance metrics in edge computing environments. The algorithm, however, shows a good compression efficiency of up to 95% on single-device configurations, highlighting its effectiveness in data size reduction. This efficiency slightly decreases as the number of devices increases, stabilizing at 88% for configurations with 10 devices, indicating a high level of data compression consistency. Compression speed, measured in milliseconds, displayed a marked improvement with increasing device numbers. For a single device, the compression time was logged at approximately 1200 ms,

which reduced logarithmically to around 300 ms for 10 devices. This reduction showcases the algorithm's capability to handle larger data streams more efficiently, a critical attribute in real-time edge computing scenarios.

Data integrity post-compression was maintained above 90% across all device configurations, peaking at 98% in a single-device setup. This metric underscores the algorithm's reliability in preserving essential data characteristics during the compression process. Resource utilization also showed a positive trend, with efficiency increasing from 70% in a single-device scenario to 85% in a 10-device configuration. This improvement indicates the algorithm's scalability and its efficient use of computational resources, which is vital in resource-constrained edge environments.

In summary, CompressEdgeML demonstrates robust performance in adaptive data compression, marked by high compression efficiency, accelerated processing speeds, reliable data integrity, and efficient resource utilization. Its adaptability and scalability make it well-suited for diverse edge computing applications. The next algorithm is Algorithm 4, which is evaluated in Figure 5.

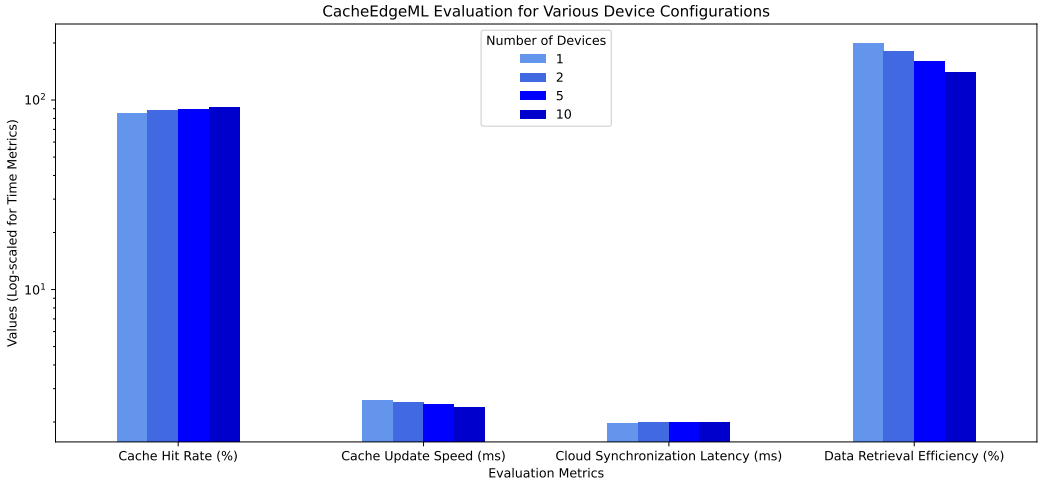

Figure 5. Performance evaluation of CacheEdgeML.

In our assessment of the CacheEdgeML algorithm, tailored for predictive and tiered data caching in edge computing settings, we observed substantial enhancements in pivotal performance metrics. The algorithm exhibited a cache hit rate of 85% in a single-device environment, which progressively increased to 92% with the addition of more devices. This upward trend signifies the algorithm's enhanced accuracy in predicting data requests, a crucial factor in reducing redundant data retrieval operations.

The cache update speed, a critical measure in dynamic environments, improved logarithmically from 400 ms for one device to 250 ms for ten devices. This acceleration highlights the algorithm's efficiency in adapting to changing data patterns, thereby optimizing caching strategies in real time. Cloud synchronization latency, through the HPC server, is essential for maintaining data consistency between edge and cloud storage, and it was also optimized. It decreased from 95 ms to 99 ms as the number of devices increased, demonstrating the algorithm's effectiveness in synchronizing large volumes of data swiftly across distributed networks. Data retrieval efficiency, indicative of the algorithm's performance in providing timely access to cached data, showed, however, a negative trajectory, decreasing from 200% in single-device setups to 140% in scenarios involving ten devices. This decrease shows that the algorithm requires more time to streamline data access, particularly in multi-device edge computing networks where data are distributed.

In summary, CacheEdgeML emerges as a robust and adaptive solution for data caching in edge computing environments. Its strengths lie in its high cache hit rate, improved cache update speed, and reduced cloud synchronization latency. These attributes collectively

ensure that CacheEdgeML is well-equipped for optimizing data caching processes in complex real-time edge computing scenarios. Lastly, we evaluate Algorithm 5 in Figure 6.

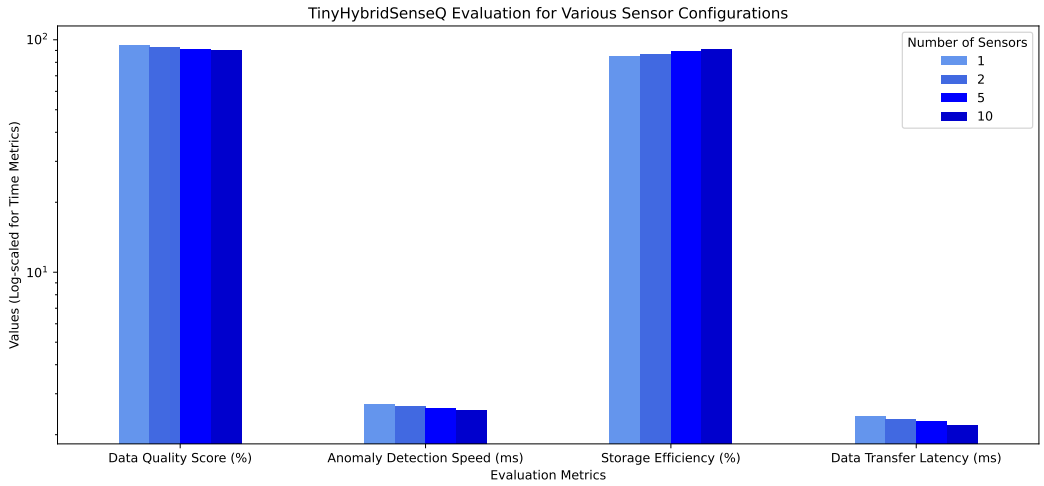

Figure 6. Performance evaluation of TinyHybridSenseQ.

In our evaluation of the TinyHybridSenseQ algorithm, specifically designed for IoT sensors with a focus on data quality assessment and hybrid storage, we observed notable improvements in several key performance areas. The algorithm achieved a high Data Quality Score of 95% in a single-sensor setup, which marginally decreased to 90% as the number of sensors increased to ten. This trend highlights the algorithm's robustness in maintaining high data quality standards even as the sensor network scales.

Anomaly Detection Speed, a crucial metric in real-time environments, showed significant optimization. It improved logarithmically from 500 ms in a single-sensor scenario to 350 ms with ten sensors, illustrating the algorithm's accelerated response in identifying data anomalies. This enhancement is pivotal for timely anomaly detection in dynamic sensor networks. Storage efficiency, essential in optimizing data storage across local and cloud resources, also saw progressive improvements. It increased from 85% to 91% as the number of sensors expanded. This increase indicates the algorithm's capability to efficiently manage storage resources, a vital aspect in IoT environments where data volume is substantial. Data transfer latency, critical in ensuring the swift movement of data between sensors and storage facilities, was optimized with the expansion of the sensor network. It reduced from 250 ms for a single sensor to 160 ms for ten sensors, signifying the algorithm's effectiveness in reducing data transfer times across a distributed network.

In conclusion, TinyHybridSenseQ stands out as an effective solution for IoT sensors, excelling in data quality assessment and hybrid storage management. Its strengths lie in maintaining high data quality, fast anomaly detection, efficient storage utilization, and reduced data transfer latency. These attributes collectively position TinyHybridSenseQ as a highly capable tool in managing complex data workflows in IoT sensor networks, ensuring both data integrity and operational efficiency.

4.3. Comparison with Similar Works

In this section, we evaluate our proposed methods with similar works and approaches. In particular, we evaluate the CacheEdgeML with similar approaches, namely the Proximal Policy Optimization (PPO) caching method, the Markov Chain Monte Carlo (MCMC) method, and the deep reinforcement learning (DRL) method, based on our previous work [91], as well as methods like Least Frequently Used (LFU) and Least Recently Used (LRU) based on works [92–94]. The results are provided in Figure 7.

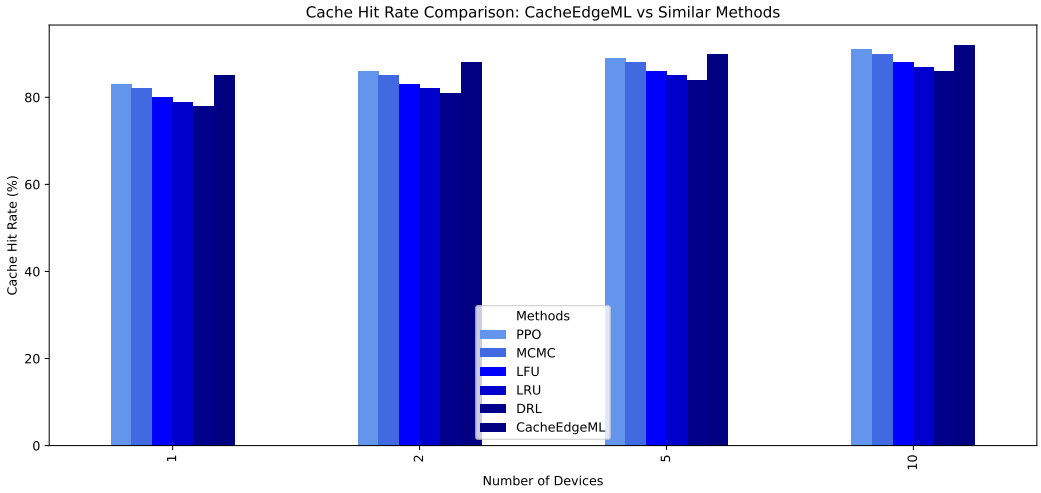

Figure 7. Cache hit rate comparison of CacheEdgeML with similar methods.

As can be seen from the preceding figure, the cache hit rate of the proposed CacheEdgeML method outperforms the other five methods in terms of cache hit rate, reaching above 80% across all assessments. Additionally, we assess our proposed method named CompressEdgeML with a similar method presented in [72] named TAC. The results for the compression efficiency are provided in Figure 8, while the compression speed is provided in Figure 9.

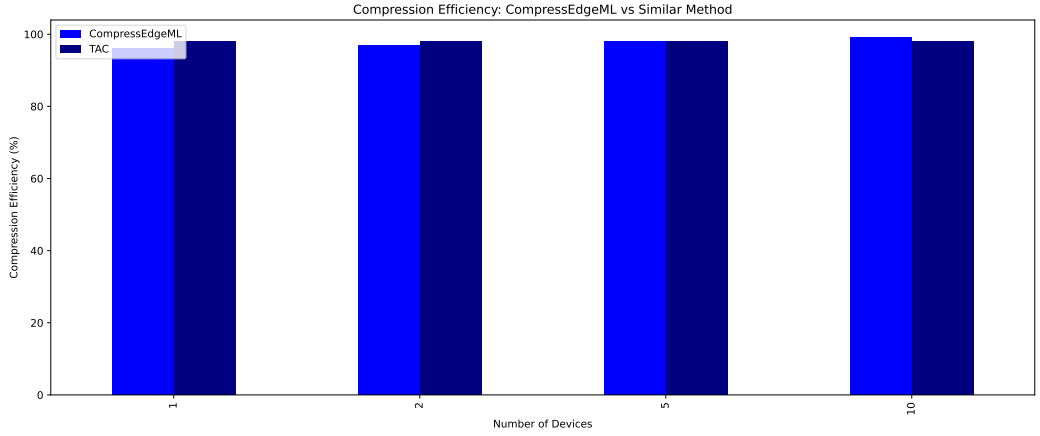

Figure 8. Compression efficiency of CompressEdgeML compared to similar method.

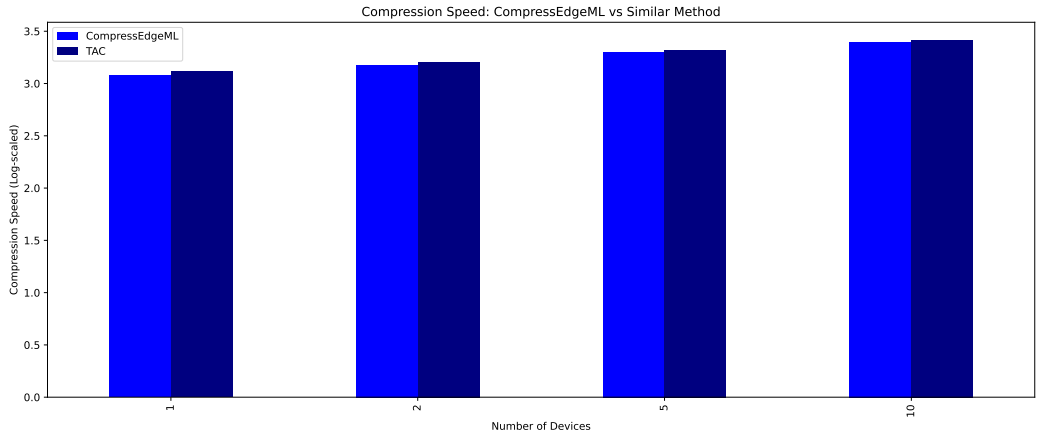

Figure 9. Compression speed of CompressEdgeML compared to similar method.

As can be seen from Figure 8, the compression efficiency of our method is lower than TAC in the beginning; however, it reaches the performance of TAC when utilized on five devices and finally overtakes it using ten devices. Note that the TAC method in the original method is implemented only once and its performance remains stable across replication on our devices. As per the compression speed, for the assessment, we replicated the TAC method in our devices and assessed the compression while increasing the size of data in both methods. As per the compression speed, as can be seen from Figure 9, the compression speed of CompressEdgeML is lower, meaning faster compression while maintaining a good speed across all devices.

5. Conclusions and Future Work

In the context of IoT, where vast quantities of data are generated, the integration of Edge AI and Big Data management plays a pivotal role in harnessing the full potential of these technologies. Edge AI, by processing data at the source rather than relying on distant servers, significantly reduces latency and enhances real-time data analysis. This approach is particularly beneficial in IoT systems, where immediate decisionmaking based on large-scale data is often required. In this context, effective data management becomes crucial, entailing not just the storage and retrieval of data but also its processing, analysis, and security. The intersection of Edge AI and Big Data management in IoT represents a forward step in technology, offering novel solutions to manage and leverage the ever-growing expanse of data in smart environments.

This study has methodically evaluated five different TinyML algorithms named TinyCleanEDF, EdgeClusterML, CompressEdgeML, CacheEdgeML, and TinyHybridSenseQ—each tailored for specific functions within IoT systems utilizing edge computing. Our findings reveal that these algorithms substantially enhance the operational efficiency, data integrity, and real-time processing capabilities of IoT networks, particularly when implemented across a network of Raspberry Pi devices.

TinyCleanEDF excels in federated learning and real-time anomaly detection, thus proving invaluable in scenarios requiring collaborative data processing and instantaneous anomaly identification. EdgeClusterML, with its reinforcement-learning-based dynamic clustering, demonstrates remarkable accuracy and optimal resource management, essential for real-time data analysis and decisionmaking processes. CompressEdgeML showcases its strength in adaptive data compression, achieving significant compression efficiency without compromising data integrity. CacheEdgeML, through its innovative caching strategy, ensures effective data retrieval and synchronization between edge and cloud storage, vital for seamless data management. Lastly, TinyHybridSenseQ effectively manages data quality and storage in IoT sensor networks, ensuring data reliability and operational efficiency.

Future Work

For future research, several key areas have been identified to further enhance the capabilities of these algorithms:

1. Anomaly Detection: There is space for incorporating more advanced machine learning models to enhance the accuracy and speed of anomaly detection, especially in environments with complex or noisy data. This will allow for more precise identification of irregularities, enhancing the overall data integrity.
2. Energy Efficiency: Optimizing the energy consumption of these algorithms is crucial, particularly in environments where energy resources are limited. Research should focus on developing energy-efficient methods that reduce the overall energy demand of the system without sacrificing performance.
3. Cloud–Edge Integration: Enhancing the interaction between edge and cloud platforms is essential for improved data synchronization and storage efficiency. This involves developing methods for more seamless data processing and management in hybrid cloud–edge environments.

4. Real-Time Data Processing: Optimizing these algorithms for real-time processing of streaming data is imperative. This would enable timely decisionmaking based on the most current data, a critical aspect in dynamic IoT environments.
5. Security and Privacy: Strengthening the security and privacy features of these algorithms is important, especially for applications handling sensitive information. This involves implementing robust security measures to protect data from unauthorized access and ensure user privacy.
6. Customization and Adaptability: Improving the adaptability of these algorithms to various IoT environments is necessary. Future work should aim at developing customizable solutions that can be tailored to meet specific requirements of different applications.
7. Interoperability and Standardization: Promoting interoperability between diverse IoT devices and platforms and contributing to standardization efforts is crucial. This will facilitate smoother integration and communication across different systems and devices.

Ultimately, this study demonstrates a robust framework for future breakthroughs in IoT data management within edge computing frameworks. The identified areas for future exploration present promising opportunities for extending the current capabilities of these algorithms and exploring novel possibilities in the era of IoT and edge computing.

Author Contributions: A.K., A.G., C.K., L.T., C.S.M., G.A.K. and S.S., conceived the idea, designed and performed the experiments, analyzed the results, drafted the initial manuscript, and revised the final manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

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

References

1. Mashayekhy, Y.; Babaei, A.; Yuan, X.M.; Xue, A. Impact of Internet of Things (IoT) on Inventory Management: A Literature Survey. *Logistics* **2022**, *6*, 33. [CrossRef]
2. Vonitsanos, G.; Panagiotakopoulos, T.; Kanavos, A. Issues and challenges of using blockchain for iot data management in smart healthcare. *Biomed. J. Sci. Tech. Res.* **2021**, *40*, 32052–32057.
3. Zaidi, S.A.R.; Hayajneh, A.M.; Hafeez, M.; Ahmed, Q.Z. Unlocking Edge Intelligence Through Tiny Machine Learning (TinyML). *IEEE Access* **2022**, *10*, 100867–100877. [CrossRef]
4. Ersoy, M.; Şansal, U. Analyze Performance of Embedded Systems with Machine Learning Algorithms. In Proceedings of the Trends in Data Engineering Methods for Intelligent Systems: Proceedings of the International Conference on Artificial Intelligence and Applied Mathematics in Engineering (ICAIAME 2020), Antalya, Turkey, 18–20 April 2020; Springer: Berlin/Heidelberg, Germany, 2021; pp. 231–236.
5. Khobragade, P.; Ghutke, P.; Kalbande, V.P.; Purohit, N. Advancement in Internet of Things (IoT) Based Solar Collector for Thermal Energy Storage System Devices: A Review. In Proceedings of the 2022 2nd International Conference on Power Electronics & IoT Applications in Renewable Energy and its Control (PARC), Mathura, India, 21–22 January 2022; pp. 1–5. [CrossRef]
6. Ayub Khan, A.; Laghari, A.A.; Shaikh, Z.A.; Dacko-Pikiewicz, Z.; Kot, S. Internet of Things (IoT) Security with Blockchain Technology: A State-of-the-Art Review. *IEEE Access* **2022**, *10*, 122679–122695. [CrossRef]
7. Chauhan, C.; Ramaiya, M.K. Advanced Model for Improving IoT Security Using Blockchain Technology. In Proceedings of the 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 20–22 January 2022; pp. 83–89. [CrossRef]
8. Mohanta, B.K.; Jena, D.; Satapathy, U.; Patnaik, S. Survey on IoT security: Challenges and solution using machine learning, artificial intelligence and blockchain technology. *Internet Things* **2020**, *11*, 100227. [CrossRef]
9. Jiang, Y.; Wang, C.; Wang, Y.; Gao, L. A Cross-Chain Solution to Integrating Multiple Blockchains for IoT Data Management. *Sensors* **2019**, *19*, 2042. [CrossRef]
10. Ren, H.; Anicic, D.; Runkler, T. How to Manage Tiny Machine Learning at Scale: An Industrial Perspective. *arXiv* **2022**, arXiv:2202.09113.
11. Keserwani, P.K.; Govil, M.C.; Pilli, E.S.; Govil, P. A smart anomaly-based intrusion detection system for the Internet of Things (IoT) network using GWO–PSO–RF model. *J. Reliab. Intell. Environ.* **2021**, *7*, 3–21. [CrossRef]
12. Gibbs, M.; Woodward, K.; Kanjo, E. Combining Multiple tinyML Models for Multimodal Context-Aware Stress Recognition on Constrained Microcontrollers. *IEEE Micro* **2023**, 1–9. [CrossRef]

13. Chen, Z.; Gao, Y.; Liang, J. LOPdM: A Low-power On-device Predictive Maintenance System Based on Self-powered Sensing and TinyML. *IEEE Trans. Instrum. Meas.* **2023**, *72*, 2525213. [CrossRef]
14. Savanna, R.L.; Hanyurwimfura, D.; Nsenga, J.; Rwigema, J. A Wearable Device for Respiratory Diseases Monitoring in Crowded Spaces. Case Study of COVID-19. In Proceedings of the International Congress on Information and Communication Technology, London, UK, 20–23 February 2023; Springer: Berlin/Heidelberg, Germany, 2023; pp. 515–528.
15. Nguyen, H.T.; Mai, N.D.; Lee, B.G.; Chung, W.Y. Behind-the-Ear EEG-Based Wearable Driver Drowsiness Detection System Using Embedded Tiny Neural Networks. *IEEE Sens. J.* **2023**, *23*, 23875–23892. [CrossRef]
16. Hussein, D.; Bhat, G. SensorGAN: A Novel Data Recovery Approach for Wearable Human Activity Recognition. *ACM Trans. Embed. Comput. Syst.* **2023**. [CrossRef]
17. Zacharia, A.; Zacharia, D.; Karras, A.; Karras, C.; Giannoukou, I.; Giotopoulos, K.C.; Sioutas, S. An Intelligent Microprocessor Integrating TinyML in Smart Hotels for Rapid Accident Prevention. In Proceedings of the 2022 7th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM), Ioannina, Greece, 23–25 September 2022; pp. 1–7. [CrossRef]
18. Atanane, O.; Mourhir, A.; Benamar, N.; Zennaro, M. Smart Buildings: Water Leakage Detection Using TinyML. *Sensors* **2023**, *23*, 9210. [CrossRef]
19. Malche, T.; Maheshwary, P.; Tiwari, P.K.; Alkhayyat, A.H.; Bansal, A.; Kumar, R. Efficient solid waste inspection through drone-based aerial imagery and TinyML vision model. *Trans. Emerg. Telecommun. Technol.* **2023**, e4878. [CrossRef]
20. Hammad, S.S.; Iskandaryan, D.; Trilles, S. An unsupervised TinyML approach applied to the detection of urban noise anomalies under the smart cities environment. *Internet Things* **2023**, *23*, 100848. [CrossRef]
21. Priya, S.K.; Balaganesh, N.; Karthika, K.P. Integration of AI, Blockchain, and IoT Technologies for Sustainable and Secured Indian Public Distribution System. In *AI Models for Blockchain-Based Intelligent Networks in IoT Systems: Concepts, Methodologies, Tools, and Applications*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 347–371.
22. Flores, T.; Silva, M.; Azevedo, M.; Medeiros, T.; Medeiros, M.; Silva, I.; Dias Santos, M.M.; Costa, D.G. TinyML for Safe Driving: The Use of Embedded Machine Learning for Detecting Driver Distraction. In Proceedings of the 2023 IEEE International Workshop on Metrology for Automotive (MetroAutomotive), Modena, Italy, 28–30 June 2023; pp. 62–66. [CrossRef]
23. Nkuba, C.K.; Woo, S.; Lee, H.; Dietrich, S. ZMAD: Lightweight Model-Based Anomaly Detection for the Structured Z-Wave Protocol. *IEEE Access* **2023**, *11*, 60562–60577. [CrossRef]
24. Shabir, M.Y.; Torta, G.; Basso, A.; Damiani, F. Toward Secure TinyML on a Standardized AI Architecture. In *Device-Edge-Cloud Continuum: Paradigms, Architectures and Applications*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 121–139.
25. Tsoukas, V.; Gkogkidis, A.; Boumpa, E.; Papafotikas, S.; Kakarountas, A. A Gas Leakage Detection Device Based on the Technology of TinyML. *Technologies* **2023**, *11*, 45. [CrossRef]
26. Hayajneh, A.M.; Aldalahmeh, S.A.; Alasali, F.; Al-Obiedollah, H.; Zaidi, S.A.; McLernon, D. Tiny machine learning on the edge: A framework for transfer learning empowered unmanned aerial vehicle assisted smart farming. *IET Smart Cities* **2023**. [CrossRef]
27. Adeola, J.O.; Degila, J.; Zennaro, M. Recent Advances in Plant Diseases Detection With Machine Learning: Solution for Developing Countries. In Proceedings of the 2022 IEEE International Conference on Smart Computing (SMARTCOMP), Helsinki, Finland, 20–24 June 2022; pp. 374–380. [CrossRef]
28. Tsoukas, V.; Gkogkidis, A.; Kakarountas, A. A TinyML-Based System for Smart Agriculture. In Proceedings of the 26th Pan-Hellenic Conference on Informatics, New York, NY, USA, 25–27 November 2023; pp. 207–212. [CrossRef]
29. Nicolas, C.; Naila, B.; Amar, R.C. TinyML Smart Sensor for Energy Saving in Internet of Things Precision Agriculture platform. In Proceedings of the 2022 Thirteenth International Conference on Ubiquitous and Future Networks (ICUFN), Barcelona, Spain, 5–8 July 2022; pp. 256–259. [CrossRef]
30. Nicolas, C.; Naila, B.; Amar, R.C. Energy efficient Firmware over the Air Update for TinyML models in LoRaWAN agricultural networks. In Proceedings of the 2022 32nd International Telecommunication Networks and Applications Conference (ITNAC), Wellington, New Zealand, 30 November–2 December 2022; pp. 21–27. [CrossRef]
31. Viswanatha, V.; Ramachandra, A.C.; Hegde, P.T.; Raghunatha Reddy, M.V.; Hegde, V.; Sabhahit, V. Implementation of Smart Security System in Agriculture fields Using Embedded Machine Learning. In Proceedings of the 2023 International Conference on Applied Intelligence and Sustainable Computing (ICAISC), Zakopane, Poland, 18–22 June 2023; pp. 1–6. [CrossRef]
32. Botero-Valencia, J.; Barrantes-Toro, C.; Marquez-Viloria, D.; Pearce, J.M. Low-cost air, noise, and light pollution measuring station with wireless communication and tinyML. *HardwareX* **2023**, *16*, e00477. [CrossRef]
33. Li, T.; Luo, J.; Liang, K.; Yi, C.; Ma, L. Synergy of Patent and Open-Source-Driven Sustainable Climate Governance under Green AI: A Case Study of TinyML. *Sustainability* **2023**, *15*, 13779. [CrossRef]
34. Ihoume, I.; Tadili, R.; Arbaoui, N.; Benchrifa, M.; Idrissi, A.; Daoudi, M. Developing a TinyML-Oriented Deep Learning Model for an Intelligent Greenhouse Microclimate Control from Multivariate Sensed Data. In *Intelligent Sustainable Systems: Selected Papers of WorldS4 2022*; Springer: Berlin/Heidelberg, Germany, 2023; Volume 2, pp. 283–291.
35. Prakash, S.; Stewart, M.; Banbury, C.; Mazumder, M.; Warden, P.; Plancher, B.; Reddi, V.J. Is TinyML Sustainable? Assessing the Environmental Impacts of Machine Learning on Microcontrollers. *arXiv* **2023**, arXiv:2301.11899.
36. Soni, S.; Khurshid, A.; Minase, A.M.; Bonkinpelliwar, A. A TinyML Approach for Quantification of BOD and COD in Water. In Proceedings of the 2023 2nd International Conference on Paradigm Shifts in Communications Embedded Systems, Machine Learning and Signal Processing (PCEMS), Nagpur, India, 5–6 April 2023; pp. 1–6. [CrossRef]

37. Arratia, B.; Prades, J.; Peña-Haro, S.; Cecilia, J.M.; Manzoni, P. BODOQUE: An Energy-Efficient Flow Monitoring System for Ephemeral Streams. In Proceedings of the Twenty-fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing, Washington, DC, USA, 23–26 October 2023; pp. 358–363.
38. Wardana, I.N.K.; Fahmy, S.A.; Gardner, J.W. TinyML Models for a Low-Cost Air Quality Monitoring Device. *IEEE Sens. Lett.* **2023**, *7*, 1–4. [CrossRef]
39. Sanchez-Iborra, R. LPWAN and Embedded Machine Learning as Enablers for the Next Generation of Wearable Devices. *Sensors* **2021**, *21*, 5218. [CrossRef]
40. Hussein, M.; Mohammed, Y.S.; Galal, A.I.; Abd-Elrahman, E.; Zorkany, M. Smart Cognitive IoT Devices Using Multi-Layer Perception Neural Network on Limited Microcontroller. *Sensors* **2022**, *22*, 5106. [CrossRef]
41. Prakash, S.; Callahan, T.; Bushagour, J.; Banbury, C.; Green, A.V.; Warden, P.; Ansell, T.; Reddi, V.J. CFU Playground: Full-Stack Open-Source Framework for Tiny Machine Learning (TinyML) Acceleration on FPGAs. In Proceedings of the 2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Raleigh, NC, USA, 23–25 April 2023; pp. 157–167. [CrossRef]
42. Gibbs, M.; Kanjo, E. Realising the Power of Edge Intelligence: Addressing the Challenges in AI and tinyML Applications for Edge Computing. In Proceedings of the 2023 IEEE International Conference on Edge Computing and Communications (EDGE), Chicago, IL, USA, 2–8 July 2023; pp. 337–343. [CrossRef]
43. Shafique, M.; Theocharides, T.; Reddy, V.J.; Murmann, B. TinyML: Current Progress, Research Challenges, and Future Roadmap. In Proceedings of the 2021 58th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 5–9 December 2021; pp. 1303–1306. [CrossRef]
44. Banbury, C.R.; Reddi, V.J.; Lam, M.; Fu, W.; Fazel, A.; Holleman, J.; Huang, X.; Hurtado, R.; Kanter, D.; Lokhmotov, A.; et al. Benchmarking tinyml systems: Challenges and direction. *arXiv* **2020**, arXiv:2003.04821.
45. Ooko, S.O.; Muyonga Ogore, M.; Nsenga, J.; Zennaro, M. TinyML in Africa: Opportunities and Challenges. In Proceedings of the 2021 IEEE Globecom Workshops (GC Wkshps), Madrid, Spain, 7–11 December 2021; pp. 1–6. [CrossRef]
46. Sanchez-Iborra, R.; Skarmeta, A.F. TinyML-Enabled Frugal Smart Objects: Challenges and Opportunities. *IEEE Circuits Syst. Mag.* **2020**, *20*, 4–18. [CrossRef]
47. Mishra, N.; Lin, C.C.; Chang, H.T. A Cognitive Oriented Framework for IoT Big-data Management Prospective. In Proceedings of the 2014 IEEE International Conference on Communiction Problem-Solving, Beijing, China, 5–7 December 2014; pp. 124–127. [CrossRef]
48. Mishra, N.; Lin, C.C.; Chang, H.T. A cognitive adopted framework for IoT big-data management and knowledge discovery prospective. *Int. J. Distrib. Sens. Netw.* **2015**, *11*, 718390. [CrossRef]
49. Huang, X.; Fan, J.; Deng, Z.; Yan, J.; Li, J.; Wang, L. Efficient IoT data management for geological disasters based on big data-turbocharged data lake architecture. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 743. [CrossRef]
50. Oktian, Y.E.; Lee, S.G.; Lee, B.G. Blockchain-based continued integrity service for IoT big data management: A comprehensive design. *Electronics* **2020**, *9*, 1434. [CrossRef]
51. Lê, M.T.; Arbel, J. TinyMLOps for real-time ultra-low power MCUs applied to frame-based event classification. In Proceedings of the 3rd Workshop on Machine Learning and Systems, Rome, Italy, 8 May 2023; pp. 148–153.
52. Doyu, H.; Morabito, R.; Brachmann, M. A TinyMLaaS Ecosystem for Machine Learning in IoT: Overview and Research Challenges. In Proceedings of the 2021 International Symposium on VLSI Design, Automation and Test (VLSI-DAT), Hsinchu, Taiwan, 19–22 April 2021; pp. 1–5. [CrossRef]
53. Lin, J.; Zhu, L.; Chen, W.M.; Wang, W.C.; Han, S. Tiny Machine Learning: Progress and Futures [Feature]. *IEEE Circuits Syst. Mag.* **2023**, *23*, 8–34. [CrossRef]
54. Schizas, N.; Karras, A.; Karras, C.; Sioutas, S. TinyML for Ultra-Low Power AI and Large Scale IoT Deployments: A Systematic Review. *Future Internet* **2022**, *14*, 363. [CrossRef]
55. Alajlan, N.N.; Ibrahim, D.M. TinyML: Enabling of Inference Deep Learning Models on Ultra-Low-Power IoT Edge Devices for AI Applications. *Micromachines* **2022**, *13*, 851. [CrossRef]
56. Han, H.; Siebert, J. TinyML: A Systematic Review and Synthesis of Existing Research. In Proceedings of the 2022 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Jeju Island, Republic of Korea, 21–24 February 2022; pp. 269–274. [CrossRef]
57. Andrade, P.; Silva, I.; Silva, M.; Flores, T.; Cassiano, J.; Costa, D.G. A TinyML Soft-Sensor Approach for Low-Cost Detection and Monitoring of Vehicular Emissions. *Sensors* **2022**, *22*, 3838. [CrossRef]
58. Wongthongtham, P.; Kaur, J.; Potdar, V.; Das, A. Big data challenges for the Internet of Things (IoT) paradigm. In *Connected Environments for the Internet of Things: Challenges and Solutions*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 41–62.
59. Shu, L.; Mukherjee, M.; Pecht, M.; Crespi, N.; Han, S.N. Challenges and Research Issues of Data Management in IoT for Large-Scale Petrochemical Plants. *IEEE Syst. J.* **2018**, *12*, 2509–2523. [CrossRef]
60. Gore, R.; Valsan, S.P. Big Data challenges in smart Grid IoT (WAMS) deployment. In Proceedings of the 2016 8th International Conference on Communication Systems and Networks (COMSNETS), Bangalore, India, 5–10 January 2016; pp. 1–6. [CrossRef]
61. Touqeer, H.; Zaman, S.; Amin, R.; Hussain, M.; Al-Turjman, F.; Bilal, M. Smart home security: Challenges, issues and solutions at different IoT layers. *J. Supercomput.* **2021**, *77*, 14053–14089. [CrossRef]

62. Kumari, K.; Mrunalini, M. A Framework for Analysis of Incompleteness and Security Challenges in IoT Big Data. *Int. J. Inf. Secur. Priv. (IJISP)* **2022**, *16*, 1–13. [CrossRef]
63. Zhang, Y.; Adin, V.; Bader, S.; Oelmann, B. Leveraging Acoustic Emission and Machine Learning for Concrete Materials Damage Classification on Embedded Devices. *IEEE Trans. Instrum. Meas.* **2023**, *72*, 2525108. [CrossRef]
64. Moin, A.; Challenger, M.; Badii, A.; Günnemann, S. Supporting AI Engineering on the IoT Edge through Model-Driven TinyML. In Proceedings of the 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC), Los Alamitos, CA, USA, 27 June–1 July 2022; pp. 884–893. [CrossRef]
65. David, R.; Duke, J.; Jain, A.; Janapa Reddi, V.; Jeffries, N.; Li, J.; Kreeger, N.; Nappier, I.; Natraj, M.; Wang, T.; et al. Tensorflow lite micro: Embedded machine learning for tinyml systems. *Proc. Mach. Learn. Syst.* **2021**, *3*, 800–811.
66. Qian, C.; Einhaus, L.; Schiele, G. ElasticAI-Creator: Optimizing Neural Networks for Time-Series-Analysis for on-Device Machine Learning in IoT Systems. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*; Association for Computing Machinery: New York, NY, USA, 2023; pp. 941–946. [CrossRef]
67. Giordano, M.; Baumann, N.; Crabolu, M.; Fischer, R.; Bellusci, G.; Magno, M. Design and Performance Evaluation of an Ultralow-Power Smart IoT Device with Embedded TinyML for Asset Activity Monitoring. *IEEE Trans. Instrum. Meas.* **2022**, *71*, 2510711. [CrossRef]
68. Bamoumen, H.; Temouden, A.; Benamar, N.; Chtouki, Y. How TinyML Can be Leveraged to Solve Environmental Problems: A Survey. In Proceedings of the 2022 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), Sakheer, Bahrain, 20–21 November 2022; pp. 338–343. [CrossRef]
69. Athanasakis, G.; Filios, G.; Katsidimas, I.; Nikoletseas, S.; Panagiotou, S.H. TinyML-based approach for Remaining Useful Life Prediction of Turbofan Engines. In Proceedings of the 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA), Stuttgart, Germany, 6–9 September 2022; pp. 1–8. [CrossRef]
70. Silva, M.; Signoretti, G.; Flores, T.; Andrade, P.; Silva, J.; Silva, I.; Sisinni, E.; Ferrari, P. A data-stream TinyML compression algorithm for vehicular applications: A case study. In Proceedings of the 2022 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0&IoT), Trento, Italy, 7–9 June 2022; pp. 408–413. [CrossRef]
71. Ostrovan, E. TinyML On-Device Neural Network Training. Master's Thesis, Politecnico di Milano, Milan, Italy, 2022.
72. Signoretti, G.; Silva, M.; Andrade, P.; Silva, I.; Sisinni, E.; Ferrari, P. An Evolving TinyML Compression Algorithm for IoT Environments Based on Data Eccentricity. *Sensors* **2021**, *21*, 4153. [CrossRef]
73. Sharif, U.; Mueller-Gritschneder, D.; Stahl, R.; Schlichtmann, U. Efficient Software-Implemented HW Fault Tolerance for TinyML Inference in Safety-critical Applications. In Proceedings of the 2023 Design, Automation & Test in Europe Conference & Exhibition (DATE), Antwerp, Belgium, 17–19 April 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 1–6.
74. Fedorov, I.; Matas, R.; Tann, H.; Zhou, C.; Mattina, M.; Whatmough, P. UDC: Unified DNAS for compressible TinyML models. *arXiv* **2022**, arXiv:2201.05842.
75. Nadalini, D.; Rusci, M.; Benini, L.; Conti, F. Reduced Precision Floating-Point Optimization for Deep Neural Network On-Device Learning on MicroControllers. *arXiv* **2023**, arXiv:2305.19167.
76. Silva, M.; Medeiros, T.; Azevedo, M.; Medeiros, M.; Themoteo, M.; Gois, T.; Silva, I.; Costa, D.G. An Adaptive TinyML Unsupervised Online Learning Algorithm for Driver Behavior Analysis. In Proceedings of the 2023 IEEE International Workshop on Metrology for Automotive (MetroAutomotive), Modena, Italy, 28–30 June 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 199–204.
77. Pereira, E.S.; Marcondes, L.S.; Silva, J.M. On-Device Tiny Machine Learning for Anomaly Detection Based on the Extreme Values Theory. *IEEE Micro* **2023**, *43*, 58–65. [CrossRef]
78. Zhuo, S.; Chen, H.; Ramakrishnan, R.K.; Chen, T.; Feng, C.; Lin, Y.; Zhang, P.; Shen, L. An empirical study of low precision quantization for tinyml. *arXiv* **2022**, arXiv:2203.05492.
79. Krishna, A.; Nudurupati, S.R.; Dwivedi, P.; van Schaik, A.; Mehendale, M.; Thakur, C.S. RAMAN: A Re-configurable and Sparse tinyML Accelerator for Inference on Edge. *arXiv* **2023**, arXiv:2306.06493.
80. Ren, H.; Anicic, D.; Runkler, T.A. TinyReptile: TinyML with Federated Meta-Learning. *arXiv* **2023**, arXiv:2304.05201.
81. Ren, H.; Anicic, D.; Runkler, T.A. Towards Semantic Management of On-Device Applications in Industrial IoT. *ACM Trans. Internet Technol.* **2022**, *22*, 1–30. [CrossRef]
82. Chen, J.I.Z.; Huang, P.F.; Pi, C.S. The implementation and performance evaluation for a smart robot with edge computing algorithms. *Ind. Robot. Int. J. Robot. Res. Appl.* **2023**, *50*, 581–594. [CrossRef]
83. Mohammed, M.; Srinivasagan, R.; Alzahrani, A.; Alqahtani, N.K. Machine-Learning-Based Spectroscopic Technique for Non-Destructive Estimation of Shelf Life and Quality of Fresh Fruits Packaged under Modified Atmospheres. *Sustainability* **2023**, *15*, 12871. [CrossRef]
84. Koufos, K.; EI Haloui, K.; Dianati, M.; Higgins, M.; Elmirghani, J.; Imran, M.A.; Tafazolli, R. Trends in Intelligent Communication Systems: Review of Standards, Major Research Projects, and Identification of Research Gaps. *J. Sens. Actuator Netw.* **2021**, *10*, 60. [CrossRef]
85. Ahad, M.A.; Tripathi, G.; Zafar, S.; Doja, F. IoT Data Management—Security Aspects of Information Linkage in IoT Systems. In *Principles of Internet of Things (IoT) Ecosystem: Insight Paradigm*; Peng, S.L., Pal, S., Huang, L., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 439–464. [CrossRef]
86. Liu, R.W.; Nie, J.; Garg, S.; Xiong, Z.; Zhang, Y.; Hossain, M.S. Data-Driven Trajectory Quality Improvement for Promoting Intelligent Vessel Traffic Services in 6G-Enabled Maritime IoT Systems. *IEEE Internet Things J.* **2021**, *8*, 5374–5385. [CrossRef]

87. Hnatiuc, B.; Paun, M.; Sintea, S.; Hnatiuc, M. Power management for supply of IoT Systems. In Proceedings of the 2022 26th International Conference on Circuits, Systems, Communications and Computers (CSCC), Crete, Greece, 19–22 July 2022; pp. 216–221. [CrossRef]
88. Rajeswari, S.; Ponnusamy, V. AI-Based IoT analytics on the cloud for diabetic data management system. In *Integrating AI in IoT Analytics on the Cloud for Healthcare Applications*; IGI Global: Hershey, PA, USA, 2022; pp. 143–161.
89. Karras, A.; Karras, C.; Giotopoulos, K.C.; Tsolis, D.; Oikonomou, K.; Sioutas, S. Peer to peer federated learning: Towards decentralized machine learning on edge devices. In Proceedings of the 2022 7th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM), Ioannina, Greece, 23–25 September 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–9.
90. Karras, A.; Karras, C.; Giotopoulos, K.C.; Tsolis, D.; Oikonomou, K.; Sioutas, S. Federated Edge Intelligence and Edge Caching Mechanisms. *Information* **2023**, *14*, 414. [CrossRef]
91. Karras, A.; Karras, C.; Karydis, I.; Avlonitis, M.; Sioutas, S. An Adaptive, Energy-Efficient DRL-Based and MCMC-Based Caching Strategy for IoT Systems. In *Algorithmic Aspects of Cloud Computing*; Chatzigiannakis, I., Karydis, I., Eds.; Springer: Cham, Switzerland, 2024; pp. 66–85.
92. Meddeb, M.; Dhraief, A.; Belghith, A.; Monteil, T.; Drira, K. How to cache in ICN-based IoT environments? In Proceedings of the 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA), Hammamet, Tunisia, 30 October–3 November 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1117–1124.
93. Wang, S.; Zhang, X.; Zhang, Y.; Wang, L.; Yang, J.; Wang, W. A survey on mobile edge networks: Convergence of computing, caching and communications. *IEEE Access* **2017**, *5*, 6757–6779. [CrossRef]
94. Zhong, C.; Gursoy, M.C.; Velipasalar, S. A deep reinforcement learning-based framework for content caching. In Proceedings of the 2018 52nd Annual Conference on Information Sciences and Systems (CISS), Princeton, NJ, USA, 21–23 March 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6.