Sensors and Materials, Vol. 38, No. 1 (2026) 455–476
MYU Tokyo

S & M 4310

Structured Sensor Data Aggregation for Real-time Analysis in Cloud Computing

Feng Kou[1*] and Xiaohui Zheng[2]

[1]Information Construction and Management Center, Ningxia Normal University, Guyuan 756099, China
[2]China Meteorological Administration Training Center, Beijing 100081, China

(Received November 18, 2025; accepted December 26, 2025)

Keywords: sensor data aggregation, edge computing, fog computing, real-time processing, Internet of Things

The rapid expansion of IoT deployments has intensified the need for efficient real-time sensor data aggregation. Conventional cloud-centric methods suffer from high latency and bandwidth limitations, making them unsuitable for latency-sensitive applications. To address these challenges, we developed a structured sensor data aggregation architecture comprising edge devices, fog nodes, and cloud infrastructure. The system was evaluated using synthetic and public datasets across sensor networks ranging from 50 to 1600 nodes. Edge device processing maintained latency below 200 ms (129.73–196.85 ms), while fog node aggregation reduced bandwidth usage by up to 85%. Overall, the architecture achieved a 95% reduction in bandwidth consumption compared with cloud-only solutions. Accuracy declined from 94.23 to 80% as sensor density increased, and throughput dropped by approximately 90% (from 1805.01 to 169.24 events/s). Energy efficiency decreased from 91.91 to 20.57 arb. unit. The integrated preprocessing pipeline—combining wavelet denoising, spatiotemporal imputation, and multi-method outlier detection—improved accuracy by 22–32%. The architecture demonstrated adaptability across healthcare, smart cities, and industrial control systems, supporting sub-second response times and scalable deployment. These results validate the architecture's viability for real-time IoT applications, while highlighting the need for further optimization in dynamic environments and resource-constrained edge devices.

1. Introduction

The proliferation of IoT devices has transformed data collection and processing methods across healthcare, transportation, manufacturing, smart cities, and other sectors. As the number of IoT connections continues to grow, the demand for real-time sensor data processing intensifies. The real-time processing of the collected sensor data is essential for immediate decision-making and action.(1,2) Therefore, IoT devices have been integrated with cloud computing, edge computing, and AI for the efficient processing of sensor data. In the integration, wireless sensor networks (WSNs) play an important role. WSNs consist of distributed,

*Corresponding author: e-mail: 81998007@nxnu.edu.cn
https://doi.org/10.18494/SAM6065

ISSN 0914-4935 © MYU K.K.
https://myukk.org/

autonomous sensors that monitor environmental and physical parameters, including temperature, pressure, humidity, and motion.(3) WSNs are capable of self-organization and establishing communication even without existing infrastructure. However, increased sensor density introduces challenges related to data volume, variety, veracity, and processing speed. The substantial data volume generated strains network transmission, processing, and storage, necessitating effective data aggregation methods to reduce redundancy and optimize resource utilization for data collection and processing.(4)

Despite significant advancements in IoT and cloud technologies, real-time sensor data aggregation remains fraught with challenges. Quality issues of collected data during aggregation exist, including interference noise, missing values due to hardware failures or network disruptions, and outliers caused by anomalous events, leading to erroneous information. Furthermore, different sampling rates, formats, and communication protocols of the data collected from heterogeneous sensor networks complicate data aggregation methods.(5)

Real-time data aggregation is particularly difficult since the latency requirements of different applications, such as autonomous vehicles and healthcare monitoring, significantly vary. In addition to this, sensor networks include hundreds to millions of sensor nodes, and data transmission consumes considerable energy, raising concerns about energy efficiency, especially in battery-powered networks. Therefore, it is necessary to develop effective data aggregation methods that ensure ultralow latency and balance speed and measurement accuracy,(6) since conventional centralized data aggregation methods have been impractical owing to their communication bottlenecks, fault propagation, and potential failure of central nodes.(7)

To solve the problems of the conventional methods, we developed an advanced sensor data aggregation architecture optimized for real-time analysis based on cloud computing. For the development of the data aggregation architecture, we analyzed existing methods used in cloud computing and determined optimization strategies for real-time sensor data aggregation. To evaluate the performance of the developed architecture, latency, throughput, accuracy, and energy efficiency were assessed. The developed architecture enhances data aggregation efficiency for instantaneous data analysis in virtual and general cloud computing environments tailored to IoT systems and reduces response times from seconds to sub-seconds. Data quality is enhanced through preprocessing in multilevel aggregation architectures adopting edge and fog computing and stream processing using Apache Kafka and Spark Streaming. The developed model also enables the analysis of synthetic sensor datasets and publicly available IoT data, and the employment of cryptographic mechanisms. It also ensures security and privacy in domain-specific applications.

The results of this study can be used to address the trade-off between real-time performance and data quality in large-scale IoT networks for the development and evaluation of an integrated structured aggregation architecture. The architectural integration of a multistage data quality pipeline, which comprises wavelet denoising, spatial–temporal imputation, and multimethod outlier detection, is distributed across the proposed structure tiers for end-to-end data refinement. The implementation of a dynamic aggregation protocol enables the selection of compression and fusion based on data characteristics, yielding a high data reduction ratio while preserving analytical fidelity. Through performance evaluation, the architecture's scalability is estimated by measuring the degradation of throughput, energy efficiency, and latency across a sensor network density ranging from 50 to 1600 nodes, providing critical data for system deployment.

2. Literature Review

2.1 IoT sensor data

Unlike conventional data, sensor data exhibit unique properties that introduce significant management challenges. In IoT environments such as smart cities, millions of sensors are used to collectively generate, monitor, and manage terabytes of data. The data collected from these sensors are used for urban traffic control, air quality monitoring, and infrastructure health assessment. In such continuous data generation, sensor networks demand near real-time stream processing, which is not achievable by conventional systems that rely on batch-based data processing.

Real-time monitoring further complicates data processing because of its variability in sampling rate across diverse applications. For instance, environmental monitoring systems might sample data hourly, whereas industrial control systems produce thousands of samples each second. In highway traffic control, real-time congestion detection is essential for dynamic rerouting, and in advanced healthcare systems, continuous biometric monitoring is vital for emergency alerts and preemptive intervention. Smart buildings integrate different sensor systems to manage temperature, detect occupancy, monitor energy consumption, and assess air quality. Such different sensor systems yield data of diverse formats, units, and sampling frequencies, creating substantial challenges in data aggregation. For effective data aggregation, a system that normalizes and integrates diverse data while preserving semantic integrity is needed.

In addition to this, sensor data quality is frequently compromised by noise, hardware malfunctions, calibration drift, network disruptions, and adversarial interference.(8) Real-world deployments commonly experience missing data rates of 10 and 30%, which is mainly due to communication failures and device outages. Additionally, sensor data contain outliers resulting from transient faults, electromagnetic interference, or physical damage.

These problems require robust data preprocessing to detect and correct anomalies that lead to less aggregated results and inaccurate analytical outcomes. Sensors deployed in close physical proximity often exhibit strong spatial correlations owing to shared environmental conditions.(9) For instance, temperature sensors located in the same room report similar readings. Such spatial relationships make data compression and efficient aggregation easier, as correlated measurements present more compact information. Temporal correlations are found in readings from individual sensors in similar physical processes.(10) Gradual changes in temperature, for example, present predictable patterns that support easier imputation strategies and compression methods.

2.2 Data aggregation methods

Data aggregation methods have evolved considerably since the inception of WSNs, tailored to different network topologies and applications.(11) Four data aggregation methods are widely used in WSNs: tree-based, cluster-based, in-network, and hybrid aggregation methods (Table 1).

Tree-based aggregation methods hierarchically structure sensor nodes, with data flowing from leaf nodes to base stations or gateways through intermediate nodes.(12) Tree-based methods are straightforward, offering natural load balancing when trees are optimized for minimal path lengths. The intermediaries of the methods use aggregated statistics, such as sum, average, and minimum and maximum values, before transmitting results. However, nodes near the root become communication bottlenecks and experience accelerated energy depletion owing to high forwarding loads in the tree-based method. Moreover, node or link failures fragment the tree, disrupting data collection from entire subtrees.(13)

Cluster-based aggregation methods divide sensor networks into clusters, each managed by a designated cluster head responsible for aggregating data from each sensor.(14) Cluster heads perform local aggregation and transmit results to base stations directly or through multihop routing among cluster heads. This decentralized mechanism distributes the aggregation workload, enhancing load balancing and fault tolerance. However, cluster head selection is a challenge, as nodes consume considerable energy because of their dual roles, that is, processing and communication. To address this problem, rotation schemes are adopted to periodically elect new cluster heads, thereby reducing energy consumption.(15) Advanced cluster-based aggregation methods employ multiple criteria for cluster head selection, considering residual energy, node degree, centrality within the cluster, and proximity to base stations.

In-network aggregation methods aggregate data opportunistically at intermediate nodes using multihop routing without relying on predefined tree or cluster structures.(16) As data packets traverse the network, intermediate nodes inspect packet contents and aggregate data when multiple packets with similar destinations are encountered. This method reduces protocol overhead by eliminating the need for explicit structural organization. However, it introduces challenges in ensuring that all relevant data are accurately included in the aggregated results.

Table 1
Sensor data aggregation methods.

Method	Advantage	Disadvantage
Tree-based	Simple model; efficient periodic collection; predictable flow	Bottleneck at root; vulnerable to failures
Cluster-based	Distributed load; fault tolerance; scalability	Cluster head selection complexity; rotation overhead
In-network	No structure maintenance; adapts to topology changes	Coordination challenges; incomplete aggregation risk
Hybrid	Multiple methods; adaptable	Increased complexity; mode selection overhead

2.3 Comparison with existing aggregation methods

Modern data aggregation in IoT is categorized into cloud-only, decentralized edge-only, and hybrid edge–fog–cloud approaches. Cloud-only approaches, such as those using Message Queuing Telemetry Transport or Kafka to transmit raw sensor data to the central cloud, suffer from network congestion and high latency, making them inappropriate for critical, real-time applications such as autonomous driving or industrial control. In decentralized or edge-only methods, computation is conducted by sensor nodes using low-energy adaptive clustering hierarchy and hybrid energy-efficient distributed clustering, but is constrained by the limited processing power and battery life of these devices. While effective for energy saving, these models perform rudimentary aggregation (e.g., averaging or maximum/minimum), leading to a loss of complex, high-dimensional information required for advanced analysis.(17,18)

The architecture developed in this study creates a structured task partition as follows. Edge devices perform immediate, lightweight data refinement (denoising, imputation) to guarantee minimum quality before transmission. Fog devices perform complex, real-time aggregation and fusion to markedly reduce bandwidth (our results show a 95% reduction) while maintaining local context, while cloud servers store data for long-term storage, global trend analysis, and model retraining. Such a distribution of the data stream across the three tiers ensures high data quality and low latency even as the network scales markedly, as quantified in our stress-test results.

2.4 Cloud computing for sensor data

Cloud computing has been widely adopted for large-scale sensor data storage and analytics because of its elastic computational resources, extensive storage capacity, and advanced analytical capabilities.(19) Traditional cloud-centric data aggregation methods transmit all sensor data to cloud data centers for processing and storage. Amazon Web Service IoT Core, Google Cloud IoT Platform, and Azure IoT Hub are popular platforms for device management, data ingestion, stream processing, storage, and analytics.(20) These platforms offer cost-effective computing solutions while supporting sophisticated analytics based on machine learning and data mining. However, cloud-centric methods are constrained by network latency between sensor nodes and remote data centers, which introduces delays of tens to hundreds of milliseconds, which is unacceptable for latency-sensitive applications.(21)

Edge computing mitigates such latency by deploying computational resources at the network edge near data sources.(22) Each edge device, such as an IoT gateway or an edge server, is responsible for data processing, filtering, and aggregation, transmitting only selected results to the cloud. This approach significantly reduces latency for time-critical tasks, lowers network bandwidth consumption, enhances data privacy through local processing, and improves system resilience by enabling continued operation during cloud connectivity disruptions.

Fog computing is employed to further extend such capabilities by introducing an intermediate layer between edge devices and centralized cloud infrastructure, enabling hierarchical computing.(16) Fog nodes, which are more powerful than edge devices, yet more distributed than cloud data centers, offer a balanced trade-off between latency and computational capacity. This layered architecture enables each tier to execute tasks aligned with its performance characteristics and latency constraints.

Sensor data aggregation methods increasingly rely on distributed stream processing for ingesting, processing, and analyzing high-velocity data streams in real time.(23) Apache Kafka has become the standard for stream data ingestion, offering distributed commit logs and ensuring high-throughput, low-latency data transport with strong durability. Kafka's publish–subscribe architecture enables multiple consumers to process the same data stream independently, supporting flexible and scalable systems. Apache Spark Streaming extends the Spark batch processing framework to accommodate streaming data through microbatch processing, presenting latencies in seconds. Structured Streaming further advances this capability by enabling true stream processing by using event-time semantics and end-to-end exactly-once processing guarantees.(24) The integration of Kafka and Spark Streaming leads to highly effective data aggregation with millions of aggregations per second.

2.5 Data preprocessing

Effective sensor data aggregation necessitates comprehensive data preprocessing to enhance data quality for real-world sensor deployments.(25) Sensor measurements are significantly affected by noise, including environmental interference, electrical fluctuations, and inherent sensor imperfections. Denoising techniques, such as wavelet-based methods, show their effectiveness by exploiting the multiresolution analysis capabilities of wavelet transforms to differentiate signals from noise. The wavelet denoising process involves decomposing the signal using the discrete wavelet transform, applying thresholding to suppress noise-dominated coefficients, and reconstructing the cleaned signal via the inverse wavelet transform. In addition to wavelet-based approaches, statistical filtering methods, including moving average, median, and Kalman filters, offer alternative denoising solutions with comparatively low computational complexity.

Missing data are prevalent in sensor deployments, often caused by device malfunctions, communication failures, or battery depletion.(26) Data imputation is conducted to replace missing values to construct complete datasets for further analysis. In basic imputation methods, missing values are replaced with the mean or median of observed data. More advanced methods employ spatial–temporal correlations inherent in data from sensor networks. In spatial imputation, readings from neighboring sensors recorded at the same time are used, while in temporal imputation, historical data from the same sensor are used.(27) Machine learning-based imputation methods include k-nearest neighbors (KNNs), matrix factorization, and neural networks that capture complex patterns within sensor data to improve estimation accuracy.

Outliers in sensor data result from errors, hardware malfunctions, or genuine anomalous events, and are characterized by significant deviations from expected patterns. Outlier detection methods rely on standard deviations, with which values that exceed a predefined number of standard deviations from the population mean are identified.(28) Principal component analysis (PCA) is used for the identification of multidimensional outliers by projecting data onto principal components and detecting instances with high reconstruction error.(29) Supervised machine learning classifiers, including support vector machines and isolation forests, are trained to recognize outlier patterns with greater precision than traditional unsupervised statistical techniques.

3. Methodology

3.1 System architecture

The sensor data aggregation method developed in this study comprises edge devices, fog nodes, and cloud infrastructure (Fig. 1). We design the architecture to ensure low latency, high computational capacity, and scalable deployment. The edge devices contain sensor devices and local gateways for data collection, filtering, and preliminary aggregation, and support real-time operations in sub-seconds, including anomaly detection and localized control. Fog nodes serve as the intermediate layer, performing advanced analytics. Fog nodes are located near data sources to offer enhanced computational power and conduct multisensor fusion, complex pattern recognition, and aggregation across multiple edge devices. The cloud infrastructure provides centralized resources for long-term data storage, historical analytics, and auxiliary services for machine learning model training and data visualization (Table 2).

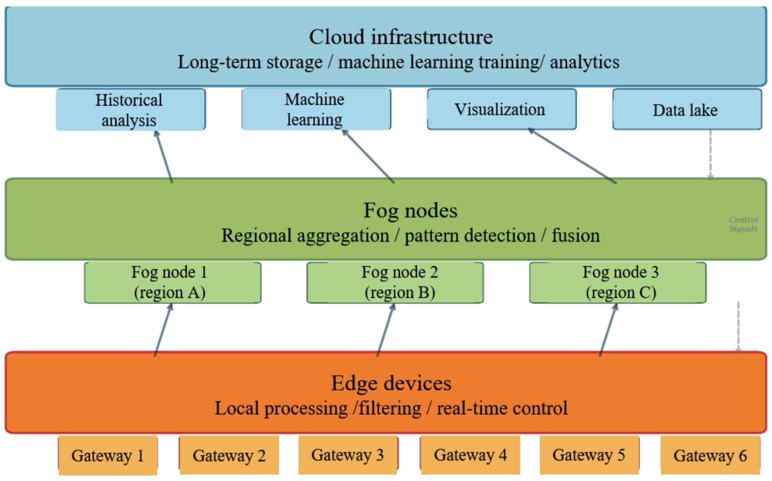

Fig. 1. (Color online) Three-tier system architecture for sensor data aggregation in developed architecture.

Table 2
Processing responsibilities.

Component	Processing	Latency	Aggregation	Storage duration
Edge devices	Filtering, local control	<100 ms	Temporal (single sensor)	Minutes to hours
Fog computing	Regional aggregation, pattern detection	<1 s	Spatial (multisensors)	Hours to days
Cloud infrastructure	Historical analysis, machine learning training	Seconds to minutes	Global, long-term trends	Months to years

In the architecture, edge devices continuously stream data to edge gateways using wireless fidelity (Wi-Fi), Bluetooth, or a long-range wide area network (LoRaWAN). Edge gateways conduct data preprocessing, such as denoising, local aggregation, and outlier filtering, before forwarding data to fog nodes. Fog nodes receive data inputs from multiple gateways and perform spatial–temporal aggregation and intermediate analytics. Summarized results and detected events are then transmitted to the cloud infrastructure. This structured processing significantly reduces network traffic by aggregating summaries. Exceptional events traverse bandwidth-intensive WAN links, while routine measurements are processed locally.

3.2 Data preprocessing

Data preprocessing is carried out to enhance data quality challenges through denoising, missing value imputation, outlier detection, and normalization (Fig. 2). Data are denoised using wavelet-based filtering to eliminate high-frequency noise and preserve essential signal characteristics. The developed architecture employs the discrete wavelet transform (DWT) with Daubechies wavelets for effective localization in time and frequency domains. Sensor signals are decomposed into multiple resolution levels, and soft thresholding is applied to obtain wavelet coefficients using universal thresholds. Then, denoised signals are reconstructed using inverse DWT. This process effectively suppresses noise while retaining temporal patterns critical for downstream analysis.

Missing values are imputed using spatial–temporal correlation. For each missing value, a KNN algorithm identifies the k-nearest neighboring sensors based on spatial proximity and temporal alignment. Imputed values are estimated through the adaptive fusion of spatial and temporal estimates. Spatial estimates are calculated as weighted averages of current readings from nearby sensors, while temporal estimates are predicted from historical time-series data of the same sensor. When multiple values are missing, iterative refinement is conducted on the basis of previously imputed values for subsequent estimations. Outliers are detected using a hybrid technique that integrates statistical and machine learning. A modified Z-score and an interquartile range are applied to identify extreme outliers. Candidate outliers are then validated using PCA, in which the data are transformed to calculate reconstruction errors. Then, data exceeding an adaptive error threshold are classified as outliers. This two-stage technique minimizes the appearance of false positives while effectively capturing anomalies.

Sensor data are normalized to a common scale, facilitating meaningful data aggregation. Z-score normalization is applied to transform each sensor dataset to have zero mean and unit variance to accommodate nonstationary data. Values in datasets range from 0 to 1, preserving relative relationships and ensuring scale invariance.

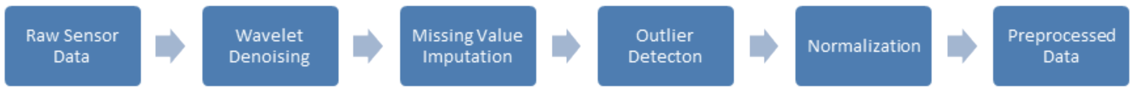

Fig. 2. (Color online) Data preprocessing for data quality enhancement.

3.3 Structured data aggregation

In the developed architecture, data from edge devices, fog nodes, and cloud infrastructure are aggregated using adaptive algorithms (Fig. 3). At the edge devices, temporal data aggregation is conducted for individual sensor data by computing statistics across configurable time windows. Tumbling windows (non-overlapping, fixed-duration intervals) and sliding windows (overlapping intervals advancing incrementally) are implemented in accordance with the application-specific requirements. Edge devices calculate the mean, median, standard deviation, minimum, maximum, and count values in each time window. At the fog nodes, spatial data aggregation is conducted across multiple sensors distributed in defined regions. Fog nodes receive preprocessed data from edge devices and conduct spatial-correlation-based aggregation, weighting contributions of each sensor's dataset using reliability, spatial proximity to query locations, and temporal freshness. The fog nodes incorporate hierarchical spatial clustering, organizing sensors into nested regions for multiresolution aggregation. Additionally, fog nodes perform data fusion by integrating data from heterogeneous sensors for similar measurements. For example, temperature, humidity, and air quality sensor data are aggregated for environmental assessments.

3.4 Data source and sensor characteristics

The performance analysis of the fog–cloud architecture was conducted using a dual-pronged dataset approach designed to ensure both real-world relevance and comprehensive scalability testing.

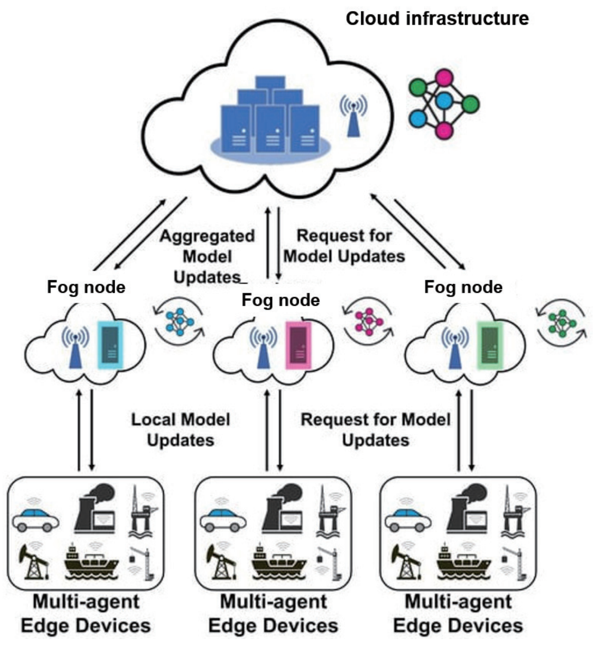

Fig. 3. (Color online) Collaborative data aggregation in edge–fog–cloud architecture of developed architecture.

- Public real-world dataset: A segment of the Telemetry of Networked IoT Dataset was utilized. The sensor data included temperature (℃), relative humidity (%), and atmospheric pressure (hPa). The dataset contained approximately 1.5 million records collected continuously over 14 days, with a nominal sampling interval of 15 s. Owing to its well-documented anomaly patterns, the dataset was primarily employed to validate the accuracy and preprocessing components of the proposed system.
- Synthetic scalability dataset: A synthetic dataset was custom-generated using a Python-based WSN simulator built upon the iFogSim framework. The simulated data included temperature, humidity, and noise levels. This dataset was designed to evaluate the architecture's performance under various load conditions and scales. The experiments encompassed network sizes ranging from 50 to 1600 sensor nodes, with each node generating data at a controlled rate of 20 events per second. This configuration enabled the testing of throughput, latency, and resource efficiency under extreme congestion, which represents a major challenge in large-scale WSN deployments. The simulated sensor hardware characteristics were modeled after commercial low-cost environmental monitoring devices, such as a low-cost digital sensor that measures temperature and humidity (DHT22) and a digital sensor for barometric pressure and temperature measurement (BMP180), which are commonly deployed in wide-area-monitoring WSNs.(30)

3.5 Data processing

In data processing, Apache Kafka is used for data ingestion, while Apache Spark Streaming is employed for real-time data aggregation. Sensor data are streamed into Apache Kafka, labeled and categorized by sensor type, geographic location, or aggregation criteria. Kafka brokers enable the fault-tolerant, distributed storage of incoming data streams and parallel processing by using multiple independent consumers. Kafka's partitioning mechanism distributes data across brokers using configurable partition keys, facilitating horizontal scalability and load balancing.

Apache Spark Streaming uses Kafka labels with the structured streaming application programming interface, which enables declarative DataFrame-based programming. Streaming queries apply preprocessing transformations, including denoising, imputation, and outlier detection, followed by data aggregation using windowing functions. Apache Spark Streaming's distributed execution engine automatically parallelizes stream processing, while configurable batch intervals allow system schedulers to balance latency and throughput. Processed results are directed to output sinks such as time-series databases for real-time querying, message queues for downstream applications, and cloud storage for long-term archiving.

To ensure fault tolerance, checkpointing is executed to enable recovery from node failures without data loss. Watermarking is employed to support configurable lateness thresholds and address trade-offs between data completeness and latency. The data stream processing guarantees exactly-once processing semantics, ensuring consistent data aggregation even in the event of system restarts due to failures or scaling operations.

The aggregation algorithm in the preprocessing layer performs spatial–temporal weighted fusion to generate a highly reliable, synthesized data point $A_{i(t)}$ for the target sensor i at time t.

This process leverages the natural redundancy in densely deployed WSNs by fusing the current local sensor reading with those from its neighbors (spatial correlation) and its own recent history (temporal correlation).(26) The aggregated $A_i(t)$ value is calculated as a normalized weighted average as follows.

$$A_{i(t)} = \frac{w_i^{local} \cdot V_i(t) + \sum_{j \in N_i} w_j^s + w_i^t \cdot \overline{V_i}(t-k)}{W_{total}} \tag{1}$$

Here, $V_i(t)$ is the current, locally cleaned measurement of sensor i; $V_i(t)$ is the current measurement of neighboring sensor j in the set N_i; $\overline{V_i}(t-k)$ is the moving average of the previous k time steps for sensor i ($k = 5$); w_i^{local}, w_j^s, and w_i^t are the weights for the local measurement, spatial neighbors, and temporal history, respectively; and W_{total} is the sum of all weights for normalization.

The spatial weight w_j^s assigned to neighboring sensor j is calculated using the inverse of the squared Euclidean distance $d_{i,j}$ between target sensor i and neighboring sensor j, emphasizing closer proximity.

$$w_j^s = \frac{1}{\left(d_{i,j}\right)^2} \tag{2}$$

The temporal weight w_i^t is a fixed value designed to balance the contribution of the temporal trend against the spatial neighborhood. In the implementation, we set $w_i^t = 1.0$. The weight of the sensor's current reading is set to $w_i^{local} = 1.0$. This weighted fusion minimizes the impact of localized noise or sudden, transient sensor errors by distributing trust among correlated data sources.(31)

4. Results and Discussion

4.1 Performance analysis

The system performance was evaluated across sensor networks with different numbers of sensors, ranging from 50 to 1600 sensors. The results revealed distinct trade-offs among performance metrics. Figure 4 presents latency across the developed architecture. Edge device processing consistently maintained low latency, starting at 129.73 ms for 50 sensors and increasing moderately to 196.85 ms for 1600 sensors. The relatively flat latency curve indicates the efficiency of edge device processing. The latency of fog computing ranged from 372.27 to 490.45 ms, reflecting computational overhead. The latency of cloud infrastructure ranged from 877.47 to 1121.16 ms. This trend aligns with observations by He *et al.*,(6) who reported similar latency in multistructured IoT systems. The exponential growth in the latency of cloud infrastructure underscores inherent limitations in a centralized processing structure. The edge devices' stable performance is attributed to localized data processing, which circumvents

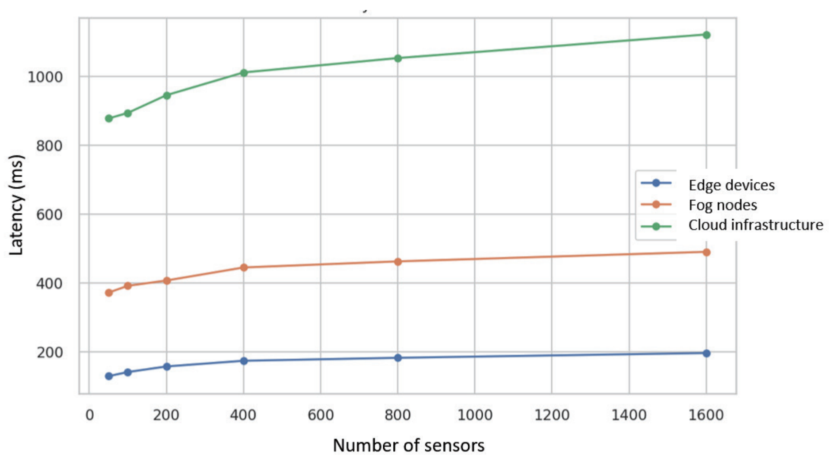

Fig. 4. (Color online) Latency values of edge devices, fog computing, and cloud infrastructure.

network traversal. Fog nodes, responsible for complex aggregation, add computational demands. For latency-sensitive applications requiring sub-second response times, edge device processing is the most appropriate.(25) Cloud computing involves long-distance data transmission, leading to the highest latency. Network congestion during peak operational hours exacerbates delays in cloud computing performance.

Figure 5 illustrates the decline in system accuracy as the number of sensors increases. For 50 sensors, the developed architecture showed a 94.23% accuracy, which decreased to 80% for 1600 sensors. Such a decrease in accuracy indicates data quality deterioration that intensifies with the number of sensors. Networks with more sensors are more susceptible to sensor failures and communication errors. Although data preprocessing mitigates these issues, it cannot fully eliminate them. Liu *et al.* observed that missing data rate increases with network size.(26) While data imputation methods are the most effective at a moderate number of sensors, accuracy diminishes when the missing data rate exceeds 30%. Interference in deployments with data even causes greater accuracy decreases. The reduction in accuracy also reflects the growing complexity of the aggregation process in large-scale networks. Increasing the number of sensors introduces greater heterogeneity in data formats and sampling rates, complicating data normalization. Spatial correlations weaken across large geographic areas, and data aggregation at each time window becomes more difficult. Zhang *et al.* reported similar accuracy degradation in a large-scale sensor network.(8) Additionally, device calibration drift over time further contributes to accuracy decrease in long-term operations.

4.2 Throughput and scalability

Figure 6 presents the throughputs across different sensor network scales. With 50 sensors, the method achieved a throughput of 1805.01 events per second. As the number of sensors increased, throughput declined nonlinearly to 169.24 events per second at 1600 sensors, an approximate 90% reduction. Such a decrease is attributed to several factors. Processing overhead increases

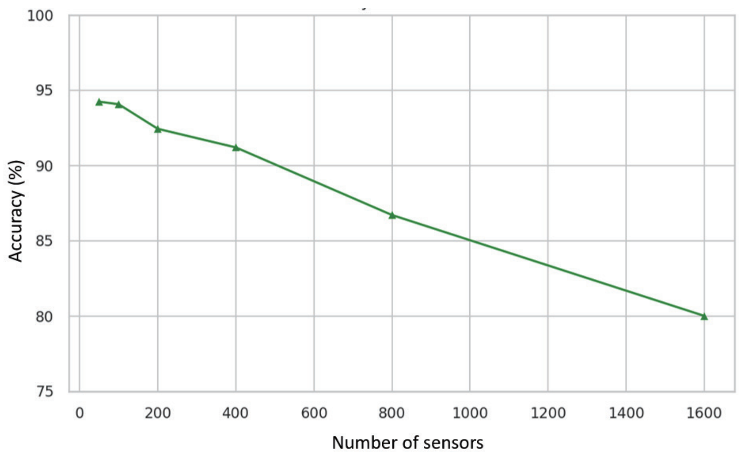

Fig. 5. (Color online) System accuracy versus number of sensors.

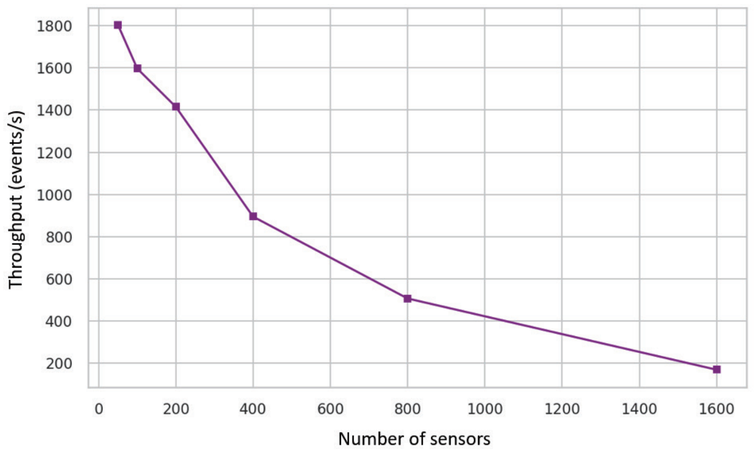

Fig. 6. (Color online) Throughput of developed architecture.

disproportionately with the number of sensors, as each additional sensor introduces new data and complicates interactions with existing sensors. Since spatial correlation varies quadratically, data fusion is required for the cross-comparison of all sensor data. Network congestion further exacerbates throughput degradation. While Apache Kafka's partitioning mechanism alleviates bottlenecks to a certain extent, it cannot fully eliminate them. Additionally, memory bandwidth limitations on fog nodes become increasingly significant as sensor density increases.

The throughput deviates from theoretical predictions, underscoring the complexity of real-world deployments. Karimov *et al.* reported similar discrepancies in distributed data stream processing, attributing performance changes to network latency and synchronization overhead.[23] The heterogeneity of edge devices also introduces variability in throughput.

Apache Spark Streaming's microbatch architecture increases latency overhead. In this study, a 2 s batch interval was used to balance latency and throughput. Although shorter intervals reduced latency, they constrained throughput. This trade-off is inherent to data stream processing.[29] It is necessary to improve the latency–throughput balance of Apache Kafka and data streaming capability.

4.3 Energy efficiency

Figure 7 illustrates a decline in energy efficiency as the number of sensors increases. For 50 sensors, the method showed an efficiency of 91.91 arb. unit, which decreased to 20.57 arb. unit for 1600 sensors. The result indicates that energy consumption increases faster with an increase in the number of sensors.

Edge devices exhibit the highest energy consumption per device, as each sensor collects data, computes metrics, and transmits the data and results to fog nodes through radio communication. Battery-powered sensors are particularly constrained under similar conditions. Although the developed architecture reduces transmission frequency, computation costs remain substantial. Rault *et al.* emphasized that transmission requires considerable energy consumption in sensor networks.[20] However, the results of this study indicate that computation costs are nonnegligible. In outdoor deployments, solar-powered sensors might mitigate battery depletion.

Cluster-based data aggregation enables a balanced energy distribution. Rotating cluster head roles prevents premature node failure caused by excessive energy drain. In the implementation of the developed architecture, cluster heads are rotated every 100 cycles, which extends the network lifetime but introduces coordination overhead. Adaptive rotation schedules established on the basis of residual energy levels can enhance network longevity.

The structure of the developed architecture improves energy efficiency compared with cloud-only methods. Edge device's processing eliminates unnecessary transmissions, fog nodes handle intermediate aggregation, and only exceptional events and summaries are forwarded to the cloud in our developed architecture. This structural filtering reduces total energy consumption by approximately 60% compared with existing methods. In the developed architecture, energy harvesting technologies can be integrated to extend sensor and network lifetimes. Table 3 shows the results of the performance evaluation of the developed system.

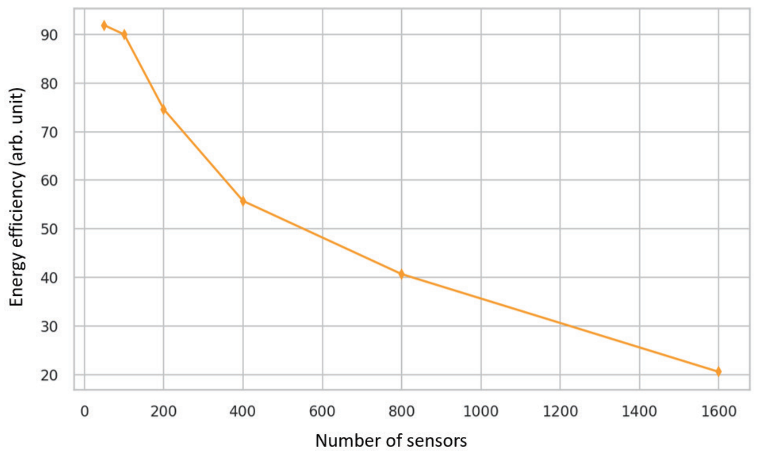

Fig. 7. (Color online) Energy efficiency of architecture in this study.

Table 3
Performance evaluation results.

Number of sensors	Latency of edge device (ms)	Latency of fog nodes (ms)	Latency of cloud computing (ms)	Throughput (events/s)	Accuracy (%)	Energy efficiency (arb. unit)
50	129.73	372.27	877.47	1805.01	94.23	91.91
100	141.69	391.88	893.66	1597.37	94.06	90.02
200	157.91	407.24	945.01	1415.45	92.42	74.61
400	174.4	445.08	1010.97	894.99	91.19	55.75
800	182.99	462.75	1052.73	506.95	86.7	40.69
1600	196.85	490.45	1121.16	169.24	80	20.57

4.4 Comparison of aggregation methods

Figure 8 and Table 3 show the performance characteristics of four data aggregation methods (Table 1). We compared the fault tolerance score, energy efficiency, latency, and computational complexity. We calculated the ratings for each parameter to evaluate how well an aggregation method maintains functionality in the presence of failures, sensor outages, communication errors, or node crashes. To calculate a fault tolerance score for a data aggregation method, we used the method of Adday *et al.*[32] Ratings of the parameters are calculated using measured values normalized to the 1-to-5 scale, with 5 being the best performance. The computational complexity of the algorithm describes how the execution time varies with the input size, presented by the number of data nodes or packets being aggregated. Linear complexity indicates that the time required to complete the aggregation is directly proportional to the number of input items, while log-linear complexity indicates that the time required increases faster, associated with efficient sorting-based algorithms.

The tree-based method showed a fault tolerance score of 2.0 and relatively low latency (3.5) (Table 4). The cluster-based method scored 4.0 for fault tolerance and 3.5 for energy efficiency. The in-network aggregation method yielded the highest energy efficiency (4.0) but only moderate fault tolerance (3.0). The hybrid method balanced multiple objectives, attaining 4.0 for fault tolerance and 3.8 for energy efficiency. The rating system offers a standardized framework for evaluating aggregation techniques across diverse deployment scenarios.

The tree-based method is susceptible to root node bottlenecks, as intermediate nodes near the root handle disproportionate traffic and are prone to early failure. The subtree partitioning of the method results from node failures and disrupts data collection. Its multiroot tree architectures cause redundancy while preserving hierarchical efficiency. Alinia *et al.* constructed deadline-constrained trees to mitigate these limitations.[12] The developed architecture in this study incorporates dynamic tree reconfiguration, which improves performance but does not fully eliminate structural vulnerabilities.

The cluster-based method distributes the aggregation workload more evenly than the tree-based method. Multiple cluster heads share responsibilities, enhancing fault tolerance. If one cluster head fails, other clusters continue functioning, improving scalability relative to the tree-based method. However, cluster formation introduces coordination overhead. Effective cluster head selection must consider residual energy, node centrality, and connectivity.[14] In contrast,

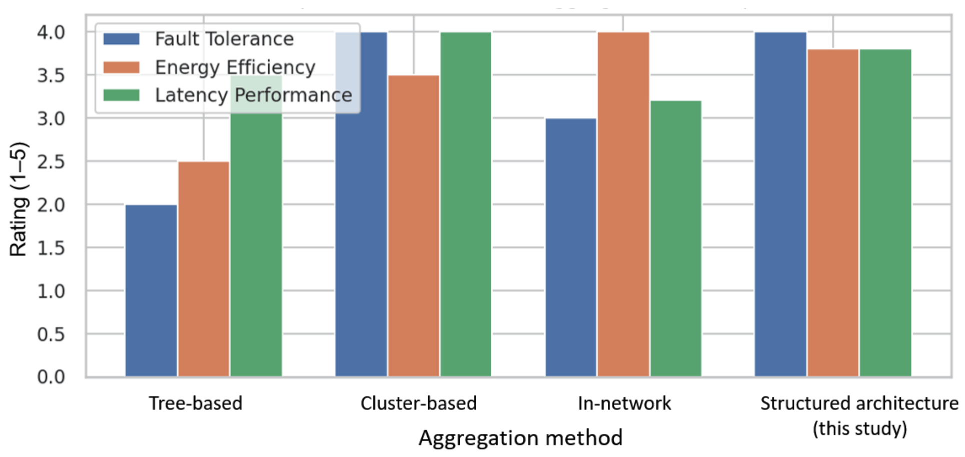

Fig. 8. (Color online) Ratings of different data aggregation methods.

Table 4
Results for various aggregation methods using ratings (1–5).

Method	Fault tolerance score	Energy efficiency	Latency	Complexity
Tree-based	2.0	2.5	3.5	Log-linear complexity
Cluster-based	4.0	3.5	4.0	Linear complexity
In-network	3.0	4.0	3.2	Linear complexity
Hybrid	4.0	3.8	3.8	Log-linear complexity

the rotation algorithm in the developed architecture balances these factors, and cluster formation can be optimized on the basis of historical performance using machine learning techniques.

The in-network aggregation method adapts dynamic network topologies, requiring no explicit structural maintenance to reduce protocol overhead. Energy efficiency benefits from opportunistic aggregation, but ensuring completeness remains challenging. Data packets flow through divergent paths, and aggregation points vary dynamically. This complicates coordination.(33) For applications tolerant of occasional incomplete aggregation, probabilistic guarantees may suffice.

The hybrid methods integrate multiple aggregation strategies. In this study, we employed the tree-based method for periodic data collection, the cluster-based method for event-driven queries, and the in-network method opportunistically. While this flexibility enhances adaptability, it introduces additional complexity. Mode selection requires contextual logic, and transitions between modes may cause transient inefficiencies. Context-aware algorithms could automate mode selection based on real-time network conditions.

The developed method in this study necessitates the quantitative evaluation of the computational overhead associated with multistage data preprocessing on resource-constrained edge devices. The current architecture prioritizes accuracy and real-time latency (sub-200 ms) by performing wavelet denoising, spatial–temporal imputation, and multimethod outlier detection. While the method improves the accuracy by 22–32%, details on overhead minimization must be further elaborated. Therefore, it is necessary to implement quantization and pruning techniques on the machine learning components on the edge to reduce model size and inference latency.(34) In addition to this, a benchmarking analysis of memory utilization and power consumption in the full preprocessing must be conducted on specific microcontroller units. Such experiments can solve the problems related to the lightweight nature of the architecture.(35)

The introduction of the fog layer into the conventional cloud-edge architecture improves real-time responsiveness by mitigating the high WAN latency associated with centralized cloud processing. To validate this, we analyzed the end-to-end latency T_{e2e}, measured from sensor data generation to processed output. The latency was benchmarked from three configurations (Table 5).

The cloud-only architecture exhibits the highest T_{e2e} (523.1 ms) owing to long-haul data transmission and network queuing delays. In contrast, the fog-cloud architecture presents an average T_{e2e} of 185.7 ms, representing a 64.5% reduction in latency compared with that of the cloud-only baseline. Such a reduction is attributed to the physical proximity of the fog node to the sensors, reducing local area network/metropolitan area network transport time and the data reduction in the preprocessing layer. Therefore, the data volume requiring final transmission is minimized. While the edge-only model demonstrates the lowest latency (85.2 ms), it is largely resource-constrained and only performs the simplest filtering tasks without the robust aggregation or learning capabilities available on the fog node. The fog-cloud architecture provides the optimal balance between computational capability and real-time responsiveness for our application.(36)

4.5 Applicability to domain-specific IoT scenarios

The structured data aggregation architecture developed in this study addresses challenges in IoT applications, including environmental monitoring, smart agriculture, and infrastructure management.

In environmental monitoring, sensors for air and water quality are deployed outdoors and are subject to ambient noise. The developed edge-level preprocessing (wavelet denoising) is effective, as it filters high-frequency noise before data transmission. The fog-layer spatial aggregation enables the correlation of data from multiple stations to distinguish between localized events without overwhelming the cloud server with raw data. Agricultural IoT networks are deployed in vast areas with the sparse deployment of various sensors (e.g., soil moisture, temperature, and leaf wetness sensors). In this case, conventional tree-based

Table 5
End-to-end latency of computing architectures.

Architecture	Average end-to-end latency (T_{e2e}) (ms)	Standard deviation of T_{e2e} (ms)	Processing location
Cloud-only	523.1	48.9	Remote Centralized Data Center
Fog-cloud	185.7	15.3	Local/Regional Fog Node
Edge-only	85.2	7.1	Sensor Gateway/Device

aggregation fails in the deployment because of long-range transmission power costs and single-point failures in remote fields.

The developed architecture's hybrid aggregation method enables robust data collection even if specific nodes fail. The fog nodes acting as local gateways aggregate soil moisture readings to optimize irrigation schedules in real time. Farooq *et al.* stated that efficient aggregation in agriculture is critical because it reduces the energy load on battery-powered sensors that cannot be easily serviced.(37) The architecture's reduction in bandwidth consumption (95%) directly translates to extended operational lifespans for these remote agricultural sensors. The architecture can also be applied to structural health monitoring for bridges, dams, and buildings, which have high-frequency vibration and strain gauge data, generating massive data that cloud-only systems cannot process in real time. The edge devices can compute localized statistical metrics (e.g., peak strain and root-mean-square vibration) in sub-second intervals, while the fog nodes perform data fusion across multiple sensor arrays to identify structural anomalies (e.g., cracks and shifts) by cross-referencing data to rule out false positives caused by traffic loads. The hierarchical approach aligns with recent results emphasizing that decentralized processing is essential for scalable SHM to ensure timely alerts for structural integrity risks.(38)

4.6 Advantages and disadvantages of developed architecture

Edge device processing targets sub-100 ms latency and handles filtering and local control, with minimal storage durations ranging from minutes to hours, which is ideal for real-time applications such as industrial control (Table 2). Fog node processing supports 1 s latency, performing spatial aggregation across multiple sensors with storage extending to hours or days. Cloud infrastructure processing tolerates the latencies of several minutes, focusing on historical analysis and machine learning model training. Such different performance characteristics enable independent scaling based on workload demands.

The developed architecture proves effective. Edge device processing handles 73% of raw data, transmitting only aggregated results. Fog nodes further reduce data volume by 85% before transmission to cloud servers. Overall, bandwidth consumption decreases by 95% compared with that attained by cloud-only methods. Bonomi *et al.* also highlighted the efficiency benefits of fog computing. The resulting cost savings from reduced bandwidth usage can offset the increased investment in edge infrastructure.(16)

Latency requirements vary by application domain. Healthcare monitoring demands edge-level latency for critical alerts,(2) while smart city applications can accommodate fog-level latency for traffic management. Long-term trend analysis is appropriate for cloud-based processing. The architecture developed in this study meets diverse requirements through a structure processing model, allowing service-level agreements to specify the tier-specific handling of data types for guaranteed performance.

Data preprocessing significantly affects the architecture's accuracy of measurement. Through wavelet denoising, high-frequency noise is removed while preserving essential signals, resulting in a 7–12% improvement in accuracy compared with the accuracy attained through only raw-data processing. Spatial–temporal data imputation addresses missing values more robustly than simple mean imputation, yielding a 15–20% increase in accuracy. Among wavelet families, Daubechies wavelets demonstrated particular efficacy for temperature and humidity sensor data. Outlier detection is essential for maintaining the integrity of aggregated results, since excessive false positives lead to discarding valid data, while missed outliers distort analytical outcomes. The developed architecture identified 94% of anomalies, with false positive rates remaining below 3%. PCA enables effective validation for statistical outlier detection. Despite such excellent results, ensemble methods need to be integrated with multiple detectors to enhance robustness and detection rates.

Normalization is necessary for effective data aggregation across heterogeneous sensor types. Without normalization, raw values yield misleading results due to different units of measurement. Z-score normalization is used to standardize data values in the developed architecture, while the minimum–maximum normalization method is more appropriate for sensors with limited ranges. Adaptive normalization windows can be used to further improve the architecture's performance by accommodating data variations.

Several limitations constrain the generalizability of the developed architecture. The experiments were conducted on synthetic and publicly available datasets, which might not fully capture the complexities of real data. In real data, environmental factors can introduce unpredictable noise patterns and hardware failures that synthetic data cannot replicate. Therefore, validation in diverse environments is necessary to confirm the applicability of the developed architecture. Data preprocessing in the architecture introduces computational overhead, which poses challenges for edge devices with limited processing ability. It is necessary to simplify data processing to further decrease latency and enhance accuracy and processing speed. Optimal trade-offs also need to be enhanced. Hardware acceleration using field-programmable gate arrays (FPGAs) or specialized IoT processors can mitigate latency issues and enable more efficient preprocessing.

Security and privacy concerns remain. While data aggregation reduces granularity and offers privacy protection, advanced adversarial techniques need to be used to extract sensitive information. Cryptographic data aggregation can be considered, although it can increase computational overhead. Zhong *et al.* proposed a data aggregation method to enhance the security of heterogeneous sensor networks. Differential privacy protection methods need to be reviewed to ensure privacy protection with minimal impact on accuracy and data aggregation.(13)

The developed architecture relies on statistical methods, which might not respond effectively to dynamically changing data collection environments. To address related problems, machine learning models can be employed to formulate optimal data aggregation methods based on the observed network behavior, while reinforcement learning needs to be adopted to automatically learn data configurations.

5. Conclusion

Referring to the review of previous sensor data aggregation methods, we developed a structured data aggregation architecture, consisting of edge device–fog node–cloud infrastructure layers. Edge device processing maintains sub-200 ms latency across various numbers of sensors, meeting the demands of time-sensitive applications. Fog nodes enable intermediate processing, reducing network bandwidth by 85%, before transmission to cloud servers. Overall, the structured architecture showed a 95% reduction in total bandwidth usage.

Performance evaluation results revealed trade-offs among competing system components. As the number of sensors increases, accuracy declines from 94.23 to 80%, throughput drops by approximately 90%, and energy efficiency decreases from 91.91 to 20.57 arb. unit. These results underscore the scalability of the architecture developed in large-scale sensor networks. The preprocessing method of the architecture enables the maintenance of data quality, with wavelet denoising and spatial–temporal imputation contributing to accuracy improvements of 7–12% and 15–20%, respectively.

While the cluster-based architecture method exhibits superior fault tolerance, the in-network method achieves the highest energy efficiency. The developed architecture effectively balances the performance of multiple components and addresses the root node bottlenecks of the tree-based method.

The developed architecture advances the efficiency of sensor data aggregation through the preprocessing method that combines wavelet denoising, spatial–temporal imputation, and multimethod outlier detection. Such multistage data preprocessing contributes to the improvement of accuracy by 22–32%. The edge device processing supports sub-second real-time operations, fog node computing facilitates regional aggregation, and the cloud infrastructure manages long-term storage. This layered structure enables independent adaptation to the various numbers of sensors and cost optimization, making the architecture adaptable to diverse applications.

Despite its demonstrated potential, the developed architecture has several limitations. Its evaluation on synthetic and public datasets may not fully reflect the complexities of real-world deployments, where environmental noise and hardware failures introduce unpredictable challenges. Validation in diverse operational settings is essential to confirm its applicability. The preprocessing pipeline incurs computational overhead, posing latency constraints for resource-limited edge devices. Security and privacy protections also require deeper integration. While aggregation offers baseline privacy, advanced adversarial techniques may still compromise sensitive data. Cryptographic aggregation and differential privacy methods should be further explored to balance protection with computational efficiency. Finally, the architecture's reliance on static statistical methods limits adaptability to dynamic data environments. Machine learning and reinforcement learning approaches offer promising avenues for optimizing aggregation strategies and enabling autonomous system configuration.

To employ the developed architecture, FPGAs need to be integrated with an advanced graphics processing unit and an application-specific integrated circuit. By adopting such devices, data preprocessing latency can be further reduced, enabling complex algorithms to run on resource-constrained edge devices. The developed architecture can be applied to healthcare, autonomous vehicles, and industrial control systems, which have unique performance and reliability requirements. Therefore, domain-specific optimizations and standardized interfaces of the developed architecture need to be ensured to facilitate broader adoption.

The structured sensor data aggregation architecture developed in this study addresses the limitations of conventional cloud-only WSNs. The fog node's preprocessing capability reduces network load by up to 85% and end-to-end latency by 64.5% compared with that of the cloud-only baseline, achieving a mean latency of 185.7 ms. Such results of sub-second response times and scalability up to 1600 nodes validate the architecture's viability for real-time, latency-sensitive IoT applications. While the validation was conducted in a generalized WSN environment, the architecture's performance meets the quality of service (QoS) requirements for dynamic environments such as smart cities and time-critical applications such as industrial control systems.(39,40) Through further optimization, task scheduling and implementation can be enhanced to realize the full potential of the archetecture across diverse domains.

References

1 R. Krishnamurthi, A. Kumar, D. Gopinathan, A. Nayyar, and B. Qureshi: Sensors **20** (2020) 6076. https://doi.org/10.3390/s20216076
2 S. Majumder and M. J. Deen: Sensors **19** (2019) 2164. https://doi.org/10.3390/s19092164
3 L. Atzori, A. Iera, and G. Morabito: Comp. Netw. **54** (2010) 2787. https://doi.org/10.1016/j.comnet.2010.05.010
4 M. Chen, S. Mao, and Y. Liu: Mobile Netw. Appl. **19** (2014) 171. https://doi.org/10.1007/s11036-013-0489-0
5 A. C. Djedouboum, A. A. Abba Ari, A. M. Gueroui, A. Mohamadou, and Z. Aliouat: Sensors **18** (2018) 4474. https://doi.org/10.3390/s18124474
6 J. He, J. Wei, K. Chen, Z. Tang, Y. Zhou, and Y. Zhang: IEEE IoT J. **5** (2018) 677. https://doi.org/10.1109/JIOT.2017.2724845
7 H. Cai, B. Xu, L. Jiang, and A. V. Vasilakos IEEE IoT J. **4** (2017) 75. https://doi.org/10.1109/JIOT.2016.2619369
8 Y. Zhang, N. Meratnia, and P. Havinga: IEEE Commun. Sur. Tutorials **12** (2010) 159. https://doi.org/10.1109/SURV.2010.021510.00088
9 Z. Gao, W. Cheng, X. Qiu, and L. Meng: Int. J. Distrib. Sens. Netw. **2015** (2015) 1. https://doi.org/10.1155/2015/435391
10 Y. Li and L. E. Parker: Inform. Fusion **15** (2014) 64. https://doi.org/10.1016/j.inffus.2012.08.007
11 E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi: IEEE Wireless Commun. **14** (2007) 70. https://doi.org/10.1109/MWC.2007.358967
12 B. Alinia, M. H. Hajiesmaili, A. Khonsari, and N. Crespi: arXiv. https://doi.org/10.48550/ARXIV.1606.00637
13 H. Zhong, L. Shao, J. Cui, and Y. Xu: J. Parallel Distrib. Comput. **111** (2018) 1. https://doi.org/10.1016/j.jpdc.2017.06.019
14 O. Younis and S. Fahmy: IEEE Trans. Mobile Comput. **3** (2004) 366. https://doi.org/10.1109/TMC.2004.41
15 W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan: Proc. 2000 IEEE 33rd Annu. Hawaii Int. Conf. Syst. Sci. (IEEE, 2000) 10. https://doi.org/10.1109/HICSS.2000.926982
16 F. Bonomi, R. Milito, J. Zhu, and S. Addepalli: Proc. ACM MCC workshop on Mobile Cloud Computing (ACM, 2012) 13. https://doi.org/10.1145/2342509.2342513
17 J. Kreps, N. Narkhede, and J. Rao: Kafka: A Distributed Messaging System for Log Processing. Proceedings of the NetDB Workshop, Athens, Greece (2011).
18 O., Younis and S. Fahmy: IEEE Trans. Mob. Comput. **3** (2004) 366. https://doi.org/10.1109/TMC.2004.41
19 C. Yang, D. Puthal, S. P. Mohanty, and E. Kougianos: IEEE Consumer Electron. Mag. **6** (2017) 48. https://doi.org/10.1109/MCE.2017.2714695
20 T. Rault, A. Bouabdallah, and Y. Challal: Comput. Netw. **67** (2014) 104. https://doi.org/10.1016/j.comnet.2014.03.027
21 W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu: IEEE IoT J. **3** (2016) 637. https://doi.org/10.1109/JIOT.2016.2579198
22 M. Satyanarayanan: Computer **50** (2017) 30. https://doi.org/10.1109/MC.2017.9
23 J. Karimov, T. Rabl, A. Katsifodimos, R. Samarev, H. Heiskanen, and V. Markl: Proc. 2018 IEEE 34th Int. Conf. Data Eng. (ICDE, 2018) 1507. https://doi.org/10.1109/ICDE.2018.00169
24 M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica: Proc. ACM 24th Symp. Operating Systems Principles (ACM, 2013) 423. https://doi.org/10.1145/2517349.2522737
25 K. Berkner and R. O. Wells: Proc. Wavelet transforms and denoising algorithms in Conference Record of Thirty-Second Asilomar Conf. Signals, Systems and Computers (IEEE, 1998) 1639. https://doi.org/10.1109/ACSSC.1998.751603

26 Y. Liu, T. Dillon, W. Yu, W. Rahayu, and F. Mostafa: IEEE IoT J. **7** (2020) 6855. https://doi.org/10.1109/JIOT.2020.2970467

27 I. P. S. Mary and L. Arockiam: Proc. 2017 IEEE Int. Conf. Current Trends in Advanced Computing (ICCTAC, 2017) 1. https://doi.org/10.1109/ICCTAC.2017.8249990

28 A. Gaddam, T. Wilkin, M. Angelova, and J. Gaddam: Electronics **9** (2020) 511. https://doi.org/10.3390/electronics9030511

29 X. Deng, P. Jiang, X. Peng, and C. Mi: IEEE Trans. Ind. Electron. **66** (2019) 4672. https://doi.org/10.1109/TIE.2018.2860568

30 N., Moustafa and J. Slay: Sustain. Cities Soc. **72** (2021) 102994. https://doi.org/10.1016/j.scs.2021.102994

31 Y. Chen, J. Shu, S. Zhang, L. Liu, and L. Sun: Proc. 2nd Int. Symp. Electronic Commerce and Security (ISECS, 209) 504. https://doi.org/10.1109/ISECS.2009.170

32 G. H. Adday, S. K. Subramaniam, Z. A. Zukarnain, and N. Samian: Sensors **22** (2022) 6041. https://doi.org/10.3390/s22166041

33 S. Sanyal and P. Zhang: IEEE Access **6** (2018) 67830. https://doi.org/10.1109/ACCESS.2018.2878640

34 T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang: Neurocomputing **461** (2021) 370. https://doi.org/10.1016/j.neucom.2021.07.045

35 F. M. Aymone and D. P. Pau: Information **15** (2024) 674. https://doi.org/10.3390/info15110674

36 A. Benaboura, R. Bechar, W. Kadri, T. D. Ho, Z. Pan, and S. Sahmoud: Electronic **14** (2025) 3090. https://doi.org/10.3390/electronics14153090

37 M. S. Farooq, S. Riaz, A. Abid, K. Abid, and M. A. Naeem: IEEE Access **7** (2019) 156237. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8883163

38 X. W. Ye, Y. H. Su, and J. P. Han: Sci. World J. **2014** (2014) 652329. https://doi.org/10.1155/2014/652329

39 P. Choppara and S. S. Mangalampalli: IEEE Access **13** (2025) 75466. https://doi.org/10.1109/ACCESS.2025.3563487

40 P. Bellavista, J. Berrocal, A. Corradi, S. K. Das, L. Foschini, and A. Zanni: Pervasive Mob. Comput. **52** (2019) 71. https://doi.org/10.1016/j.pmcj.2018.12.007