Real-time adaptive cleaning of iot sensor data using machine learning noise classification and rule-based refinement

Viktoriia Zhebka[1,†], Svitlana Shlyanchak[2,†], Svitlana Popereshnyak[1,3,†] and Dmytro Nishchemenko[1,*,2,†]

[1] State University of Information and Communication Technologies, 7 Solom'yanska str., Kyiv, 03110, Ukraine
[2] Volodymyr Vynnychenko Central Ukrainian State University, Shevchenko Street, 1, Kropyvnytskyi, 25000, Ukraine
[3] National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", 37, Prospect Beresteiskyi, Kyiv, 03056, Ukraine

Abstract

Real-time Internet of Things sensor data is frequently corrupted by mixed noise (outliers, drift, constant values), degrading data quality. Conventional cleaning methods often lack adaptivity for such heterogeneous noise. This paper introduces URTCA, a novel real-time cleaning architecture for univariate time series. URTCA employs a machine learning classifier (Random Forest) on sliding window features to identify distinct noise types. This classification, refined by a rule-based check for low-variance states, drives an adaptive strategy module that selects appropriate cleaning operators (e.g., imputation, smoothing, pass-through). Experiments using simulated noise on real temperature data demonstrated URTCA's superior cleaning accuracy, achieving ~14% and ~34% lower Root Mean Squared Error (RMSE) compared to Rolling Median and Kalman Filter baselines, respectively. URTCA offers a robust, classification-driven adaptive solution for enhancing the reliability of real-time IoT data streams.

Keywords

data cleaning, real-time, internet of things, noise classification, machine learning

1. Introduction

The proliferation of the Internet of Things (IoT) has led to an exponential increase in connected devices, especially within smart home and industrial contexts, generating vast streams of sensor data [4]. This data, typically univariate time series like temperature or energy usage, holds immense potential but is often plagued by quality issues [18]. Raw sensor data streams are frequently compromised by errors stemming from sensor malfunctions, environmental factors, or transmission glitches [2, 3]. Common data quality problems include missing values, isolated point outliers, continuous errors like sensor drift or constant segments, and general background noise [3]. These issues can significantly degrade the performance of downstream applications, making data quality management crucial, particularly in domains like industrial IoT and smart grids.

Many IoT applications require real-time processing, rendering traditional batch cleaning methods insufficient [7]. The high velocity and continuous nature of IoT data demand online approaches where cleaning occurs as data arrives. Furthermore, merely detecting errors or discarding data is often inadequate; applications like forecasting or control systems typically require a complete and corrected time series [8, 9]. Therefore, real-time data correction, replacing erroneous values with

*MoMLeT-2025: 7th International Workshop on Modern Machine Learning Technologies, June, 14, 2025, Lviv-Shatsk, Ukraine
* Corresponding author.
† These authors contributed equally.
viktoria_zhebka@ukr.net (V. Zhebka); shlanchaksveta@gmail.com (S. Shlianchak); spopereshnyak@gmail.com (S. Popereshnyak); dima.nishchemenko@gmail.com (D. Nishchemenko)
0000-0003-4051-1190 (V. Zhebka); 0000-0001-9893-5709 (S. Shlianchak); 0000-0002-0531-9809 (S. Popereshnyak); 0009-0006-1781-4109 (D. Nishchemenko)

plausible estimates, is a critical necessity [7]. Failure to adequately clean and correct data can propagate errors, leading to flawed analyses and unreliable system behavior.

This paper focuses on addressing these challenges through a specific methodology: classification-driven adaptive correction for real-time cleaning of univariate IoT time series data. This approach involves classifying detected noise or anomalies into distinct types (e.g., outlier, drift, constant value) using machine learning techniques [1], and then dynamically selecting or parameterizing a correction strategy best suited to the identified noise type [5]. This contrasts with simpler methods like generic smoothing filters (e.g., EWMA), which can distort valid data patterns, or non-adaptive methods that apply a uniform repair irrespective of the error's nature [9]. The underlying principle is that different error types necessitate different, tailored correction techniques for optimal results [2, 3]. The central challenge is thus to accurately diagnose the specific error type in real-time and apply an appropriate correction that removes the artifact while preserving genuine data patterns, navigating the trade-off between minimal data modification and necessary correction [8].

To address this challenge, we propose URTCA, a novel unified real-time cleaning architecture. Our method utilizes a machine learning classifier [1] to identify multiple noise types within sliding windows of sensor data and employs an adaptive strategy module, refined by rule-based logic, to apply targeted cleaning actions. This paper details the architecture, implementation, and evaluation of the proposed method, demonstrating its effectiveness in handling mixed noise types compared to existing approaches.

The remainder of this paper is structured as follows: Section 2 reviews the relevant literature on real-time IoT data cleaning, focusing on noise classification and adaptive correction. Section 3 presents the architecture and components of the proposed URTCA system. Section 4 describes the experimental setup, including data generation, comparison methods, and evaluation metrics. Section 5 presents and discusses the experimental results, including cleaning performance and classification accuracy. Finally, Section 6 concludes the paper and outlines directions for future work.

2. Related works

Research addressing data quality issues in real-time IoT streams has yielded various techniques for anomaly detection, data imputation, and stream processing [9, 18]. This literature review concentrates on the specific method of classification-driven adaptive cleaning applied to univariate IoT time series.

The core idea behind classifying noise before correction is that different error types (e.g., spikes, drift, constant segments) have distinct characteristics and causes, warranting tailored repair strategies [2, 3]. Applying a universal correction method risks either ineffective repair or distortion of valid data [9]. Therefore, identifying the specific noise type allows for a more targeted and potentially more accurate correction that better preserves signal integrity [8].

Machine learning models like Random Forests (RF), Decision Trees (DT), and Support Vector Machines (SVM) are potential candidates for automating this classification task, given their ability to learn complex patterns from features [1]. However, a review of the literature suggests that the explicit use of such standard classifiers for multi-class noise type identification (distinguishing outlier vs. drift vs. constant segment etc.) as a preliminary step to adaptive correction is not widely documented [9]. Much existing work focuses on binary anomaly detection (normal vs. anomalous) using techniques like LSTM, Autoencoders, statistical tests on model residuals, or distance-based methods [3, 16], rather than categorizing the anomaly's nature. Systems combining ML with data cleaning exist [10, 11], but specific details on multi-type noise classification are often limited.

A key challenge for supervised noise classification is effective feature engineering [20]. Potential features derived from sliding windows include statistical moments, trend indicators, constancy measures, frequency domain information, or model-based residuals [15]. While some methods implicitly use such features (e.g., IQR statistics, model residuals, decomposition components), designing a feature set to robustly distinguish diverse noise types in real-time remains difficult. Furthermore, the requirement for accurately labeled training data, where each point or segment is

tagged with a specific noise type, presents a major bottleneck [19], explaining the prevalence of unsupervised or semi-supervised approaches in anomaly detection research [3].

Adaptive correction aims to dynamically select or parameterize the cleaning action based on the diagnosed noise type, moving beyond one-size-fits-all approaches [5]. Hypothetically, point outliers could be handled by local imputation, drift might require detrending, constant segments might need model-based interpolation, and high background noise could be addressed by adaptive smoothing filters [12].

While the surveyed literature shows examples of adaptivity, it often differs from the target paradigm. Methods like Iterative Minimum Repairing (IMR) [8] and ARX/ANN cleaning algorithms [15] adapt by iteratively refining a single underlying model based on previous repairs. Other approaches adapt based on statistical triggers like IQR or skewness thresholds, or contextual information from neighboring sensors [18]. Tools like cleanTS [10] automate sequential steps but rely on user selection rather than dynamic classification-based adaptation. Systems like Cleanits [11] suggest integrated strategies but lack detail on classification-driven adaptation. Adaptive smoothing methods like ASPA [12] adjust parameters based on signal characteristics but typically target background noise without prior multi-class classification. Thus, the explicit linkage where a classifier's output label (e.g., "Drift") directly selects a specific correction algorithm (e.g., "Detrending") appears uncommon [9]. Rule-based logic and thresholds, sometimes learned dynamically (e.g., using ELM for speed constraints) [17], offer a complementary way to incorporate adaptivity and domain knowledge.

Implementing sophisticated cleaning in real-time IoT environments faces constraints like sequential data arrival, low latency demands, limited computational resources (especially on edge devices), and high data volume [4, 7]. Distributed architectures, potentially using edge computing for initial processing and dynamic task scheduling, are proposed to manage these constraints [4, 18]. Several online algorithms exist, including streaming outlier detectors (TsOutlier) [16], adaptive smoothers (ASPA) [12], iterative methods with efficient updates (IMR) [8], stream constraint checkers (SCREEN) [13], rolling window statistics, and dynamic constraint predictors [17]. However, efficient integration of the entire pipeline (feature extraction, ML classification, adaptive correction selection, execution) in real-time remains a major challenge [7], requiring careful design and potential algorithm simplification or hardware acceleration.

Practical deployment often requires integrated systems combining detection, correction, and other functionalities [20]. Existing frameworks like Cleanits [11], IMR [8], ARX/ANN algorithms [15], and cleanTS [10] each have different focuses (industrial integration, iterative repair, automation). While valuable, these systems generally do not appear to fully implement the specific paradigm of explicit multi-class noise classification directly driving the selection among diverse, tailored correction algorithms in real-time [9].

Evaluating real-time cleaning involves assessing correction accuracy (RMSE, MAE against ground truth) [8], impact on downstream tasks (e.g., forecasting accuracy) [15], and computational efficiency (latency, throughput). Obtaining ground truth often requires using real-world data with synthetically injected, labeled errors [20]. While studies report performance improvements using various advanced methods [8, 11, 15], a major challenge is the lack of standardized benchmarks specifically designed to evaluate classification-driven adaptive cleaning, including metrics for the classification step itself and the benefit of adaptation [9].

In summary, while the field addresses real-time IoT data cleaning with various ML and statistical techniques [18], a specific gap exists concerning methods that explicitly classify multiple distinct noise types using ML [1] and use this classification to dynamically select tailored correction strategies in real-time [9]. Key research gaps include the development and validation of such multi-class noise classifiers for time series, designing architectures that effectively link classification to adaptive correction under real-time constraints, creating appropriate benchmarks, and advancing real-time feature engineering. Addressing these gaps, potentially through hybrid approaches combining ML and rules and leveraging edge computing [4], is crucial for advancing the state-of-

the-art in robust and reliable IoT data cleaning. This research aims to contribute to filling these gaps by proposing and evaluating URTCA, a unified real-time cleaning architecture.

3. Architecture and components

To address the challenges of cleaning mixed noise types in real-time univariate IoT sensor data streams, we propose the URTCA. The core principle is classification-driven adaptive cleaning, refined by rule-based logic to handle low-noise states effectively. The method processes data sequentially, making decisions based on information available within a sliding window of recent data points.

The architecture consists of the following key components, operating iteratively as new data points arrive (see Figure 1):

1. Sliding Window Buffer: A fixed-size buffer (implemented using a deque for efficiency) maintains the most recent W data points (raw or imputed). In our experiments, a window size W=10 was used, corresponding to 10 minutes of data. This buffer provides the necessary historical context for feature extraction and cleaning actions.
2. Feature Extraction Module: For each new data point that fills the buffer, a set of statistical features is calculated from the current window's data. These features aim to capture characteristics indicative of different noise types. The features calculated include:
    a. Basic moments: Mean, Variance, Standard Deviation.
    b. Order statistics: Minimum, Maximum, Range.
    c. Robust statistics: Interquartile Range (IQR), Median Absolute Deviation (MAD).
    d. Relative variation: Coefficient of Variation (CV = Std Dev / Mean).
    e. Change indicators: Mean Absolute Difference between consecutive points.
    f. Missing data indicators: Count and Fraction of NaNs within the window. NaN values within the window are handled during feature calculation to ensure robust feature generation.
3. Noise Classifier: A machine learning classifier, specifically a Random Forest model in our implementation, is employed to predict the dominant noise type present within the current window based on the extracted features. The classifier is trained offline using labeled data from a separate training set containing synthetically generated noise examples. In our final configuration, the classifier was trained to distinguish between four primary noise categories relevant to the cleaning task: Outlier, ConstantValue, Drift, and GeneralNoise. It does not explicitly predict a 'Clean' state.
4. Rule-Based Refinement Module: Recognizing that distinguishing true low-noise states from low-amplitude 'GeneralNoise' can be challenging for the classifier, a rule-based check is applied after the classification step. If the classifier predicts GeneralNoise, but the calculated variance of the current window falls below a predefined threshold, the prediction is overridden.
5. Adaptive Strategy Selection & Cleaning Operators: Based on the effective label (either the classifier's prediction or the 'Clean' label assigned by the rule-based refinement), a specific cleaning strategy is selected and applied to the current data point:
    a. Clean (Effective Label): The original data point is passed through without modification. This occurs if the rule-based refinement identifies a low-variance state.
    b. Outlier: A robust Z-score (calculated using window median and MAD) is checked for the current point. If it exceeds a threshold (e.g., 3.5), the point is considered an outlier and replaced by the window's median value; otherwise, it is passed through.
    c. ConstantValue: The point is replaced by the median of the last few points (e.g., 3) in the buffer, assuming recent valid points are more reliable than the potentially anomalous constant value.Drift: The point is replaced by the mean of a slightly larger

number of recent points (e.g., 5) in the buffer, providing smoothing to counteract the drift trend.

d. GeneralNoise: (Applied only if variance > threshold). The point is replaced by the mean of the last few points (e.g., 3) in the buffer, providing gentle smoothing
e. Missing Value (NaN Handling): If the incoming point itself is NaN, it is imputed using the median of the recently cleaned values stored by the cleaner, ensuring that the imputation relies on the best available estimates of the recent true signal level.

This sequence of buffering, feature extraction, classification, rule-based refinement, and adaptive strategy application allows the proposed method to dynamically adjust its cleaning approach in real-time based on the diagnosed characteristics of the incoming data stream.

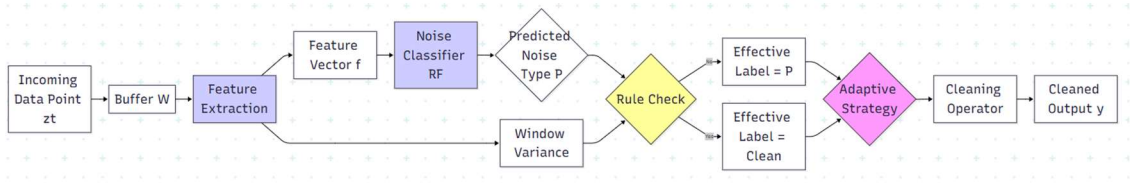

Figure 1: Flowchart of the proposed URTCA real-time cleaning architecture.

4. Experimental setup

To evaluate the performance of the proposed URTCA architecture, we conducted experiments using a real-world dataset with synthetically injected noise, comparing our method against relevant baselines under simulated real-time conditions.

4.1. Data and preprocessing

The base dataset originates from sensors deployed in a single home located in northeastern Mexico, collected over 14 months (November 5, 2022, to January 5, 2024) at one-minute intervals. The full dataset comprises 605,260 samples across 19 variables related to energy consumption and weather conditions. For this study, we focused on a single representative univariate time series: indoor temperature (temp). A continuous one-week segment (10,080 data points) from January 1, 2023, 00:00:00 to January 7, 2023, 23:59:00 was extracted from the original dataset to serve as the clean "ground truth" data for our experiments. This segment exhibited typical diurnal temperature variations.

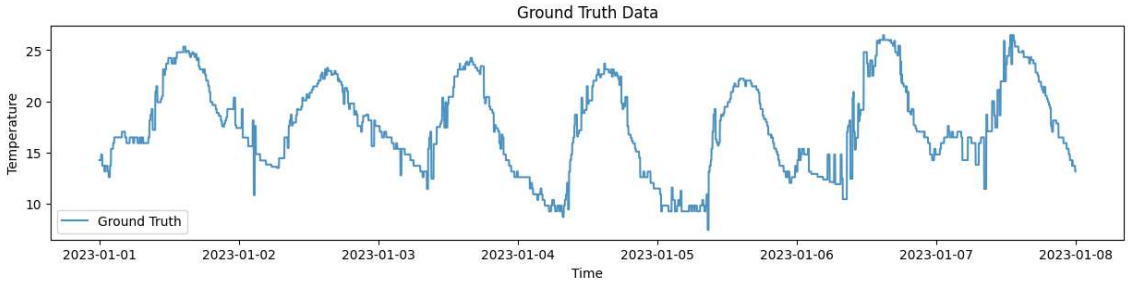

Figure 2: Flowchart of the real set part.

To create a realistic test set with known error types and ground truth, we injected a controlled mixture of synthetic noise onto the clean ground truth data using the following procedure (with a fixed random seed for reproducibility):

1. Point Outliers: Approximately 3% of randomly selected points were designated as outliers. Noise drawn from a normal distribution scaled by 5 times the ground truth standard deviation was added to these points.
2. Constant Value Segments: 25 segments with random durations between 20 and 50 minutes were selected, and the values within these segments were replaced by the constant value observed at the start of each segment.
3. Drift Segments: 20 segments with random durations between 90 and 200 minutes were selected. A linear drift (with a maximum slope factor relative to the overall standard deviation) was added to the original values within these segments.
4. General Noise: Gaussian noise with a standard deviation equal to 5% of the ground truth standard deviation was added to a fraction (60%) of the remaining clean points. Points not affected by specific noise types or this step remained labeled 'Clean'.
5. Missing Values: Approximately 5% of the data points across the entire series were randomly selected and replaced with NaN values.

The approximate distribution of underlying noise types (before NaN overlay) in the final noisy dataset was: GeneralNoise (~36%), Drift (~29%), Clean (~24%), ConstantValue (~8%), Outlier (~2%).

The ground truth and corresponding noisy datasets were split chronologically into training (first 70%, 7056 points) and testing (last 30%, 3024 points) sets. The Random Forest classifier for the proposed method was trained only on features extracted from the training set. All performance evaluations were conducted only on the unseen testing set.

4.2. Comparison methods

The performance of the proposed method URTCA was compared against the following methods, all operating under the same real-time simulation constraints.

The raw noisy test dataset serves as the baseline to quantify the overall noise level and the minimum expected error.

Rolling Median is a common non-adaptive filtering technique. It replaces each point with the median value calculated over a sliding window of the preceding W noisy data points. The window size W was set to 10, matching the window size used by the proposed method.

Kalman Filter (Local Level) is a standard model-based filtering technique suitable for real-time processing. We implemented a simple local level model assuming the true temperature follows a random walk with Gaussian process noise (variance Q) and is observed with Gaussian measurement noise (variance R). The observation noise variance R was estimated from the standard deviation of the injected Gaussian noise ($R \approx 0.025$). The process noise variance Q was set heuristically to a small value ($Q = 1e\text{-}4$) assuming relatively smooth temperature changes minute-to-minute. The filter recursively predicts the state and updates it based on the current observation.

4.3. Evaluation metrics

The performance of all cleaning methods was evaluated using the following metrics on the test set.

Root Mean Squared Error (RMSE) measures the square root of the average squared difference between the cleaned/estimated values and the ground truth values. This metric is ensitive to large errors.

Mean Absolute Error (MAE): Measures the average absolute difference between the cleaned/estimated values and the ground truth values. Unlike RMSE, it is less sensitive to outliers.

The percentage reduction in RMSE and MAE achieved by the proposed method compared to the baseline methods was calculated by taking the difference between the metric value of the baseline method and the metric value of the proposed URTCA method, dividing that difference by the metric value of the baseline method, and then multiplying the result by 100.

To understand the behavior of the internal classification mechanism, we compared the effective label assigned by the URTCA method at each step (e.g., 'Outlier', 'Drift', 'Clean_Rule' mapped to

'Clean', etc.) against the true underlying noise type injected into the test data. Metrics included: Overall Accuracy, Precision, Recall, F1-score (per class, weighted, and macro averages), Confusion Matrix.

The average processing time per data point (in milliseconds) was measured for each method during the simulation run to compare their computational efficiency.

5. Experimental results

This section presents the results of the real-time cleaning simulation conducted on the synthetic test dataset, comparing the proposed URTCA against the baseline methods: Rolling Median, Kalman Filter, and the original noisy data (Noisy).

5.1. Cleaning Performance Comparison

The primary evaluation focused on the accuracy of data cleaning, measured by Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) against the ground truth test data. Table 1 summarizes the performance metrics for all evaluated methods.

Table 1
Comparison of Cleaning Performance Metrics on the Test Set

Method	RMSE	MAE
URTCA	0.862444	0.279588
Rolling Median	1.003315	0.365204
Kalman Filter	1.301516	0.841901
Noisy	3.208738	0.841901

As shown in Table 1 and visualized in Figure 3, the proposed URTCA method achieved the lowest RMSE (0.862) and MAE (0.280), indicating the highest overall cleaning accuracy among the tested methods.

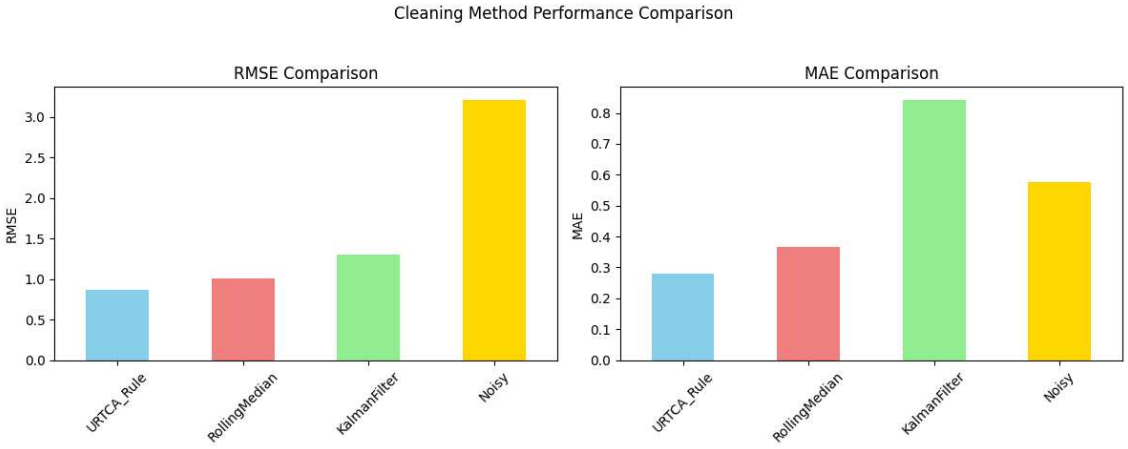

Figure 3: Bar chart comparing RMSE (left) and MAE (right) for different cleaning methods.

It significantly outperformed the simple Rolling Median filter (RMSE 1.003, MAE 0.365) and the model-based Kalman Filter (RMSE 1.302, MAE 0.842). Both URTCA and Rolling Median provided substantial improvements over the Noisy baseline (RMSE 3.209, MAE 0.575), while the implemented Kalman Filter, although better than Noisy, struggled more with the mixed noise types present in the data.

To quantify the advantage of the proposed method, the percentage improvement in RMSE and MAE was calculated relative to the baseline methods. Table 2 presents these improvements.

Table 2
Percentage Improvement of URTCA_Rule over Baseline Methods

URTCA over Method	RMSE	MAE
Rolling Median	14.04%	23.44%
Kalman Filter	33.74%	66.79%

URTCA demonstrated a notable ~14% reduction in RMSE and ~23% reduction in MAE compared to the Rolling Median. The improvement over the Kalman Filter was even more pronounced, with reductions of ~34% in RMSE and ~67% in MAE. These results highlight the benefit of the adaptive, classification-driven approach over both simple filtering and a standard model-based filter when dealing with heterogeneous noise types.

Figure 4 provides a qualitative comparison, illustrating how the different methods handled a segment of the noisy test data compared to the ground truth. Visual inspection suggests that URTCA effectively smooths general noise, corrects outliers, and adapts to shifts or constant segments more accurately than the Rolling Median or Kalman Filter, which tend to either oversmooth or fail to adequately correct certain anomaly types.

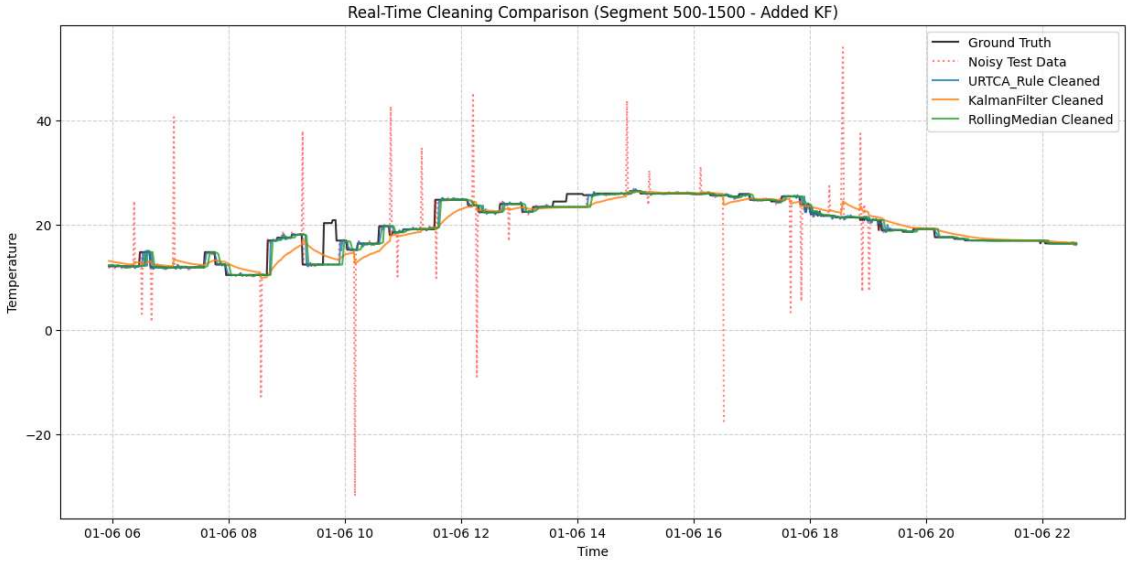

Figure 4: Example segment comparing Ground Truth, Noisy Data, and cleaned output from URTCA, Rolling Median, and Kalman Filter.

5.2. Internal Classification Performance

While the primary goal is cleaning accuracy, understanding the performance of the internal noise classification mechanism within URTCA provides valuable insights. We evaluated the accuracy of the effective label assigned by the method (after the rule-based refinement) compared to the true underlying noise type injected into the test data.

The overall accuracy of assigning the correct effective label was approximately 59.7%. Figure 4 visualizes the confusion matrix.

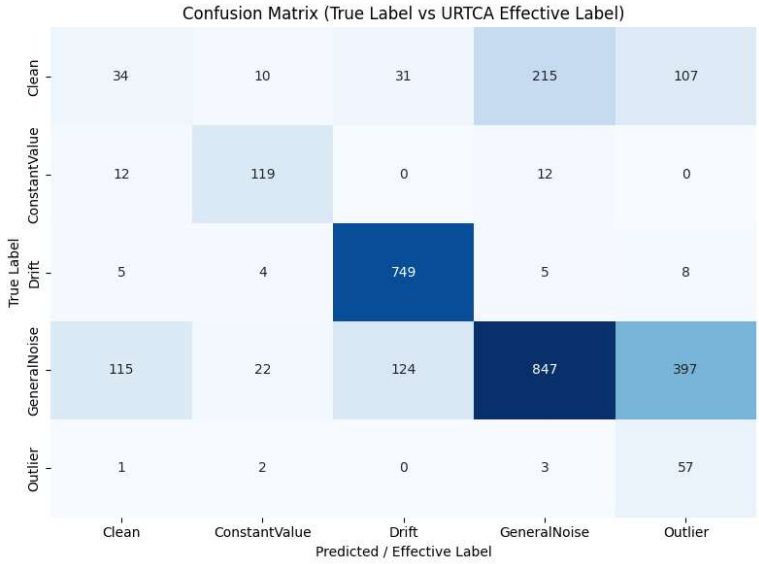

Figure 5: Confusion Matrix for URTCA (True Label vs. Effective Label) on the Test Set.

The classification results indicate effective identification of Drift (F1=0.880) and ConstantValue (F1=0.780) segments. The method also demonstrates high recall (0.838) for Outlier detection, meaning it successfully flags most true outliers. However, the precision for Outlier (0.100) and Clean (0.204) is low, indicating that many segments identified as these types were actually instances of other noise, primarily GeneralNoise. GeneralNoise itself had relatively low recall (0.532), often being misclassified as Outlier or Clean (via the rule-based override).

Despite these imperfections in the underlying classification, particularly the difficulty in distinguishing low-level GeneralNoise from truly Clean segments or certain Outliers, the overall URTCA architecture achieved superior cleaning performance (Table 1, Figure 3). This suggests that the combination of the adaptive strategy selection (applying tailored cleaning for correctly identified Drift, ConstantValue, and high-confidence Outliers) and the rule-based handling of low-variance states effectively compensates for the classification ambiguities, leading to robust and accurate data correction.

5.3. Computational Cost

The average processing time per data point was measured during the simulation. URTCA required approximately 32.6 ms per point, primarily due to feature extraction and classifier inference. This is significantly higher than the Rolling Median (~0.06 ms) and the Kalman Filter implementation (~1.3 ms in our test, though this can vary), but remains feasible for many real-time smart home applications involving data sampled at intervals of seconds or minutes, such as the 1-minute data used here.

6. Conclusions

This paper addressed the critical challenge of cleaning univariate time series data from IoT sensors in real-time, particularly when faced with a mixture of different noise and anomaly types. Traditional methods often struggle in such scenarios, either by applying overly simplistic filtering or relying on model assumptions violated by diverse error patterns. We proposed URTCA, a novel architecture based on classification-driven adaptive cleaning, refined by rule-based logic.

Our experimental results, based on simulations using real-world temperature data corrupted with synthetic mixed noise, demonstrate the effectiveness of the proposed approach. URTCA significantly outperformed both a standard filtering technique (Rolling Median) and a common model-based approach (Kalman Filter) in terms of cleaning accuracy, achieving approximately 14% and 34% lower RMSE, respectively. This superior performance highlights the primary contribution of this work:

demonstrating that an adaptive strategy, guided by machine learning-based classification of noise types and augmented with rules for handling low-noise states, can lead to more accurate and robust real-time data cleaning compared to non-adaptive or simpler model-based methods when dealing with complex, heterogeneous noise profiles typical of IoT sensor data. The analysis of the internal classification mechanism revealed that while distinguishing certain noise types remains challenging, the overall adaptive architecture effectively compensates, yielding high-quality cleaned data suitable for reliable downstream applications.

The significance of this work lies in providing a practical and demonstrably effective real-time cleaning methodology tailored to the complexities of IoT sensor data. By moving beyond generic filtering or detection, URTCA enables more nuanced data correction, which is crucial for improving the reliability of analytics, control systems, and decision-making processes built upon sensor streams. This contributes to unlocking the full potential of IoT data in various domains.

However, this study has limitations. The evaluation relied on synthetically generated noise; further validation on diverse real-world datasets with ground truth or well-characterized errors is necessary. The internal noise classifier, while effective within the overall system, showed limitations in discriminating certain noise types, suggesting room for improvement through advanced feature engineering or alternative classification models. The current implementation is univariate, and extending the architecture to handle multivariate time series, considering inter-channel correlations, remains an important future step. Furthermore, the method's parameters (window size, thresholds) were set heuristically and may require tuning for different datasets or noise characteristics. Lastly, while feasible for minute-level data, the computational cost needs consideration for higher frequency streams or severely resource-constrained edge devices.

Future work should focus on enhancing the noise classification module, potentially exploring semi-supervised learning or more sophisticated feature extraction techniques. Extending the framework to multivariate data cleaning is a key direction. Investigating methods for automatic parameter optimization and exploring model compression or hardware acceleration techniques to reduce computational overhead for edge deployment would also be valuable. Finally, applying and evaluating the method in specific real-world IoT application domains (e.g., energy forecasting, industrial predictive maintenance) will further validate its practical utility.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] Breiman, L. Random Forests. *Machine Learning* 45, 5–32 (2001), doi:10.1023/A:1010933404324.

[2] V. Hodge, J. Austin, A Survey of Outlier Detection Methodologies, Artif. Intell. Rev. 22.2 (2004) 85–126. doi:10.1023/b:aire.0000045502.10941.a9..

[3] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection, ACM Comput. Surv. 41.3 (2009) 1–58. doi:10.1145/1541880.1541882.

[4] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of Things (IoT): A vision, architectural elements, and future directions, Future Gener. Comput. Syst. 29.7 (2013) 1645–1660. doi:10.1016/j.future.2013.01.010.

[5] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. 46.4 (2014) 1–37. doi:10.1145/2523813.

[6] R. E. Kalman, A New Approach to Linear Filtering and Prediction Problems, J. Basic Eng. 82.1 (1960) 35–45. doi:10.1115/1.3662552.

[7] K. Yasumoto, H. Yamaguchi, H. Shigeno, Survey of Real-time Processing Technologies of IoT Data Streams, J. Inf. Process. 24.2 (2016) 195–202. doi:10.2197/ipsjjip.24.195.

[8] A. Zhang, S. Song, J. Wang, P. S. Yu, Time Series Data Cleaning: From Anomaly Detection to Anomaly Repairing, Proc. VLDB Endow. 10.10 (2017) 1046–1057. doi:10.14778/3115404.3115410.

[9] X. Wang, C. Wang, Time Series Data Cleaning: A Survey, IEEE Access 8 (2020) 1866–1881. doi:10.1109/access.2019.2962152.
[10] M. K. Shende, A. E. Feijóo-Lorenzo, N. D. Bokde, cleanTS: Automated (AutoML) Tool to Clean Univariate Time Series at Microscales, Neurocomputing (2022). doi:10.1016/j.neucom.2022.05.057.
[11] X. Ding, H. Wang, J. Su, Z. Li, J. Li, H. Gao, Cleanits: a data cleaning system for industrial time series, Proc. VLDB Endow. 12.12 (2019) 1786–1789. doi:10.14778/3352063.3352066.
[12] Rong K., Bailis P. ASPA: Adaptive Smoothing for Streaming Time Series, Proc. VLDB Endow. 10.11 (2017) 1358–1369. doi:10.14778/3137628.3137645.
[13] Song, S., Zhang, A., Wang, J., & Yu, P.S., SCREEN: Stream Data Cleaning under Speed Constraints, Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data.
[14] R. Ahmad, E. H. Alkhammash, Online Adaptive Kalman Filtering for Real-Time Anomaly Detection in Wireless Sensor Networks, Sensors 24.15 (2024) 5046. doi:10.3390/s24155046.
[15] H. N. Akouemo, R. J. Povinelli, Data Improving in Time Series Using ARX and ANN Models, IEEE Trans. Power Syst. 32.5 (2017) 3352–3359. doi:10.1109/tpwrs.2017.2656939.
[16] Yogita, D. Toshniwal, A Framework for Outlier Detection in Evolving Data Streams by Weighting Attributes in Clustering, Procedia Technol. 6 (2012) 214–222. doi:10.1016/j.protcy.2012.10.026.
[17] Yin, M., and Yue, K. Time Series Data Cleaning Method Based on Optimized ELM Prediction. JIPS, vol. 13, no. 2, 2017, pp. 432-445. doi:10.3745/JIPS.04.0268.
[18] A. Karkouch, H. Mousannif, H. Al Moatassime, T. Noel, Data quality in internet of things: A state-of-the-art survey, J. Netw. Comput. Appl. 73 (2016) 57–81. doi:10.1016/j.jnca.2016.08.002.
[19] B. Zhu, C. He, P. Liatsis, A robust missing value imputation method for noisy data, Appl. Intell. 36.1 (2010) 61–74. doi:10.1007/s10489-010-0244-1.
[20] An, N., Ding, Y., and Zhao, H. Statistical feature analysis and preprocessing assisted artificial neural network for cleaning multi-type concurrent anomalies in time series data. Proc. ISCAS, 2024. doi: 10.1109/ISAS61044.2024.10552599