International Journal of Basic and Applied Sciences, 15 (2) (2026) 79-93

International Journal of Basic and Applied Sciences

Website: www.sciencepubco.com/index.php/IJBAS
https://doi.org/10.14419/ezf67m88
Research paper

A Deep Learning Approach for Time Series Data Correction: Integrating Autoencoder-Based GANs and Correlation Analysis

Sohyeon YUN *, Han-Joon KIM

Department of Electrical and Computer Engineering University of Seoul, Seoul, Korea
**Corresponding author E-mail: khj@uos.ac.kr*

Received: December 27, 2025, Accepted: February 28, 2026, Published: April 26, 2026

Abstract

This paper presents a novel correction framework for multivariate time series data that enhances data quality by integrating Autoencoder-based Generative **Adversarial** Networks (GANs) with correlation-aware analysis. With the widespread **adoption** of deep learning models for sensor-driven applications such as quality control and demand forecasting, data anomalies caused by sensor faults and network errors have emerged as a critical challenge, often degrading model performance. In **multivariate** time series, anomaly correction is particularly difficult because erroneous values must be distinguished from legitimate temporal variations while preserving inter-variable dependencies and temporal consistency. To address this challenge, we propose a data correction method that combines deep learning–based anomaly detection with correlation-driven correction. An Autoencoder-based GAN is **employed** to identify anomalous patterns in multivariate time series, while a window relevance matrix is introduced to guide precise correction. This matrix captures complex relationships among variables by jointly incorporating dynamic time warping and Pearson correlation coefficients within sliding windows. By **leveraging** both temporal alignment and statistical dependency, the proposed approach performs anomaly correction that maintains structural coherence across variables. Extensive experiments conducted on diverse multivariate time series datasets demonstrate that the corrected data consistently improves predictive performance compared to raw data. These results indicate that the proposed method effectively enhances data quality and model reliability, offering a robust solution for anomaly correction in multivariate time series applications.

Keywords: *Multivariate Time Series Data; Deep Learning; GAN; Anomaly Detection; Data* ***Correction****; Data Quality.*

1. Introduction

The proliferation of Internet of Things (IoT) technology has revolutionized data collection and analysis across diverse sectors, including smart factories, smart cities, financial services, and energy management [1]. Through networks of interconnected sensors, IoT systems continuously gather vast quantities of real-time data [2], facilitating comprehensive system monitoring [3], anomaly detection, predictive maintenance, and rapid incident response. These systems generate time series data—sequential recordings that capture the evolving states of systems and environments. In manufacturing applications, for example, crucial parameters such as equipment temperature, humidity, and pressure are systematically analyzed to optimize operational performance and develop predictive models for preventing equipment failures [4], [5].

Time series data frequently contains anomalies stemming from various data collection challenges [6]. These anomalies—values that deviate significantly from normal patterns—can arise from sensor malfunctions, data transmission errors, or environmental interference. A temperature sensor might record impossibly high readings during a malfunction, or network interruptions could lead to gaps in data storage [7]. When such anomalous data is incorporated into training datasets, it compromises both data reliability and model performance, making anomaly detection and correction crucial for maintaining system integrity [8], [9]. The challenge becomes particularly complex with multivariate time series data, where multiple variables interact dynamically [10]. Effective correction of multivariate data requires simultaneous consideration of both temporal patterns and inter-variable relationships—a requirement that many existing detection and correction methods fail to address, as they are either designed for simpler univariate data or cannot adequately capture the complex interactions among variables.

In this study, we introduce a novel approach to detecting and correcting anomalies in multivariate time series data by combining an LSTM (Long Short-Term Memory)-based VAE-GAN (Variational Autoencoder Generative Adversarial Networks) model [11] with a time-aware window relevance matrix. Our method augments the conventional window relevance matrix through a novel integration of Dynamic Time Warping (DTW) and Pearson correlation coefficients, enabling more sophisticated capture of complex relationships among variables. This enhanced approach transcends basic anomaly detection by implementing corrections that preserve essential data relationships, thereby improving data quality and significantly boosting the performance of deep learning-based predictive models. We validated our method

through extensive experiments using diverse public datasets, including SWaT (Secure Water Treatment) [12], WADI (Water Distribution) [13], HAI (HIL-based Augmented ICS) [14], and SKAB (Skoltech Anomaly Benchmark). The experimental results demonstrated substantial improvements in predictive performance compared to raw data, highlighting our method's potential to enhance data quality and model reliability across industrial applications.

The rest of the paper is organized as follows. Section 2 presents a comprehensive review of anomaly detection and correction research, covering existing detection methods, key concepts of generative models employed in our experiments, and the process of generative model-based anomaly detection. This section also examines current trends in data correction methodologies. Section 3 provides a detailed exposition of our proposed correction method for multivariate time series data. Section 4 presents our experimental validation, demonstrating the effectiveness of our approach through comparative analysis with existing correction methods using predictive model performance. Section 5 concludes the paper by synthesizing our key findings and outlining promising directions for future research.

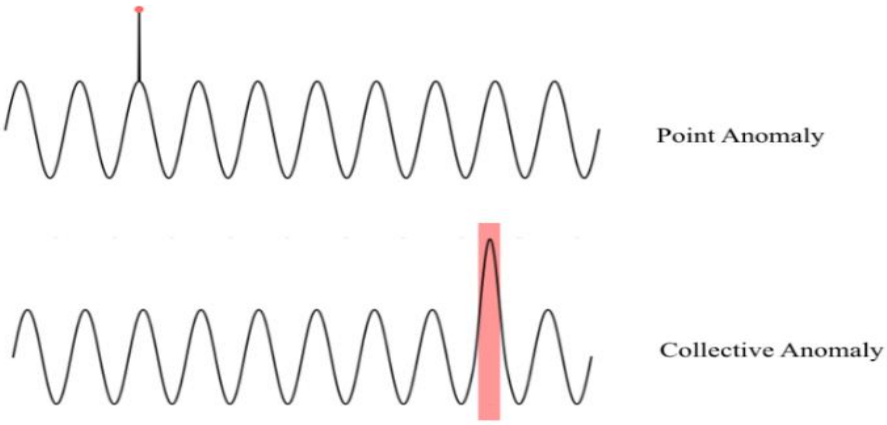

Fig. 1: Two Types of Anomalies in Time Series Data.

2. Related Work

2.1. Anomaly detection

Anomalies in time series are observations that deviate from normal patterns and are commonly categorized as point anomalies (single abnormal points) and collective anomalies (abnormal behavior across a sequence) [15]. Figure 1 visually illustrates examples of point and collective anomalies in time series data. Such anomalies degrade data quality and can reduce the reliability of downstream analytics and predictive models; therefore, anomaly detection is a key step for improving data quality and model robustness [16].

Early approaches relied on statistical and proximity-based techniques (e.g., mean/variance analysis, density-based, and distance-based methods) [17–19], but their effectiveness and scalability can be limited as data dimensionality and temporal complexity increase. Recently, deep learning-based methods have become popular because they can capture nonlinear temporal dynamics and interdependencies in multivariate time series [20]. Representative architectures include: (i) LSTM-based predictors, which flag anomalies via deviations between observed and predicted values [21]; (ii) autoencoders, which detect anomalies using reconstruction error learned from normal data (shown in Figure 2) [22]; and (iii) GAN-based methods, where the discriminator learns normal data distributions and treats deviations as anomalies [23].

In this study, we adopt an LSTM-based VAE-GAN that integrates sequential modeling, variational latent representations, and adversarial learning to perform anomaly detection as part of the proposed correction framework.

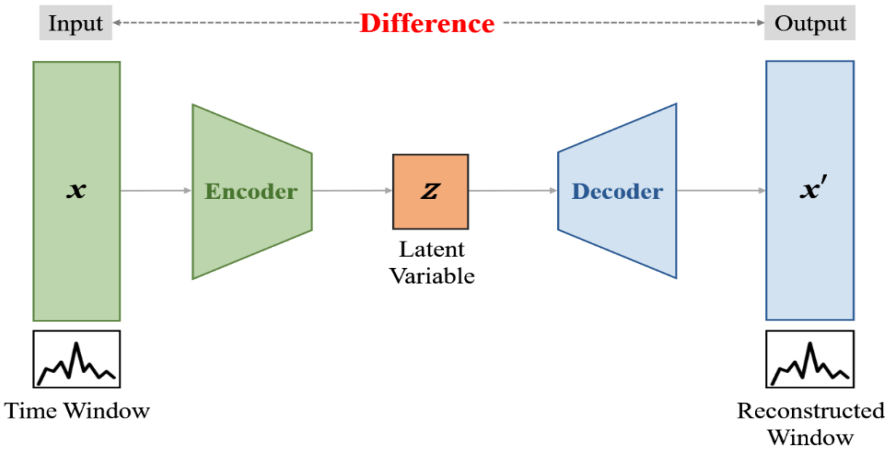

Fig. 2: Architecture of Autoencoder-based Anomaly Detection.

2.2. Data correction

Compared to anomaly detection, research on correcting detected anomalies in time series is relatively less extensive. Nevertheless, anomaly correction is practically important because anomalies can degrade data reliability and, in turn, reduce the accuracy of downstream analysis and predictive models. Existing correction approaches range from simple statistical replacements to learning-based methods.

Traditional strategies include mean/median-based replacement, which is computationally efficient but often fails to preserve local temporal patterns (e.g., moving average smoothing) [24], and anomaly removal, which reduces anomalous influence but may discard informative structure and harm performance when anomalies are frequent. Learning-based correction methods such as SVR replace anomalous values with model predictions learned from normal data, but they can be limited in high-dimensional or multivariate settings [25].

More recently, deep learning-based correction has gained attention. LSTM-based models leverage temporal dependencies to estimate plausible values from surrounding context [26], while autoencoders restore inputs by reconstructing normal patterns learned from clean data, often preserving multivariate structure more effectively than simple statistical methods [27].

In this study, we propose an LSTM-based VAE-GAN correction framework that exploits temporal dynamics in multivariate time series and aims to outperform existing correction techniques. Table 1 summarizes representative correction approaches and position the advantages of the proposed method.

Table 1: Comparison of Time Series Anomaly Correction Approaches.

Approach	Method	Key Limitation	How our method address it
Statistical replacement	mean/median, moving average	Often distorts local temporal patterns; limited for complex dynamics	Uses a generative model to produce pattern-consistent corrections
Removal-based	Delete anomalies	Loses information; breaks temporal continuity; degrades when anomalies are frequent	Corrects (replaces) anomalous windows instead of removing them
Regression-based	SVR/predictive imputation	Scalability issues in high-dimensional multivariate data	LSTM-based latent modeling captures temporal dynamics; correlation-aware relevance exploits inter-variable relations
Deep learning-based	LSTM/autoencoder reconstruction	May ignore inter-variable dependency structure or produce overly smooth reconstructions	Combines temporal alignment (DTW) and dependency similarity (Pearson correlation) to select informative reference windows

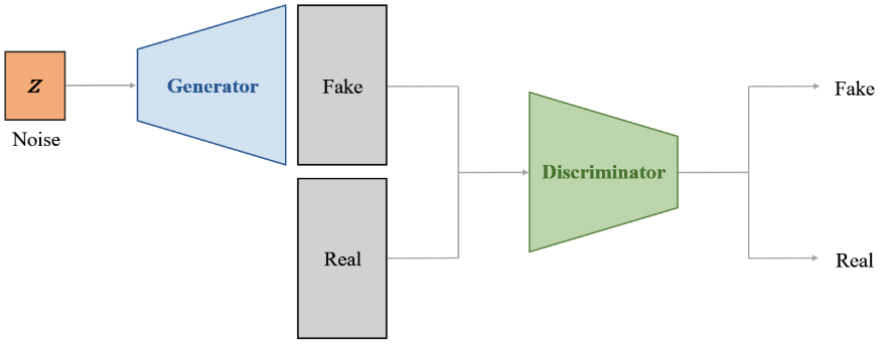

Fig. 3: Architecture of Basic GAN.

2.3. Generative adversarial network (GAN)

In this study, we adopt a VAE-GAN model for anomaly detection and data correction; therefore, we briefly summarize the core idea of GANs. A GAN consists of two networks—a generator G and a discriminator D—trained in an adversarial manner to model the data distribution [28]. The generator maps a noise vector z to a synthetic sample $G(z)$, while the discriminator outputs the probability $D(x)$ that an input x is real. Training is formulated as the minimax objective:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{x\sim p_z(z)}[\log(1 - D(G(z)))] \quad (1)$$

Through this adversarial process, G learns to generate samples that resemble real data, and D learns to distinguish real from generated samples. At convergence, the generated samples become difficult to differentiate from real ones, enabling GANs to serve as effective generative models for tasks such as anomaly detection and data correction (as seen in Fig. 3).

3. The proposed Methods

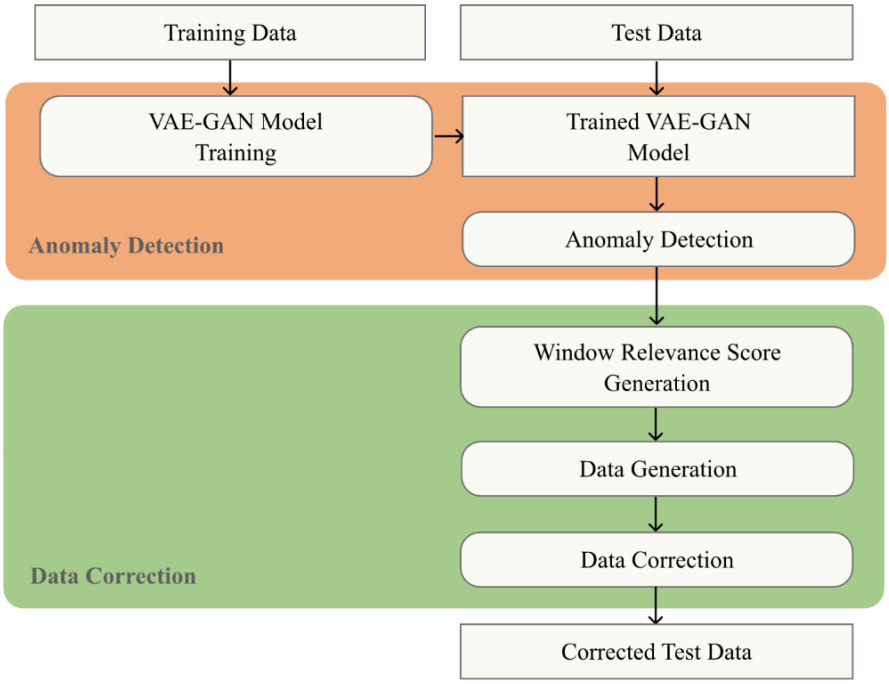

Fig. 4: Overall Process of the Proposed Data Correction Method.

A VAE-GAN is trained on normal data and applied to test data to detect anomalous windows; relevant reference windows are then selected via a window relevance score and used by the generator to replace anomalous windows, producing corrected test data.

Figure 4 illustrates the overall process of the proposed data correction method. As previously mentioned, this study employs a VAE-GAN model for anomaly detection and data correction in multivariate time series data. The proposed method consists of two main stages.

First, anomaly detection is performed as a prerequisite for correcting anomalies in multivariate time series data. The VAE-GAN model is trained on normal data to learn typical patterns, and the trained model is then used to identify anomalous patterns in the test data. This stage detects anomalies based on distributional differences between normal and anomalous data.

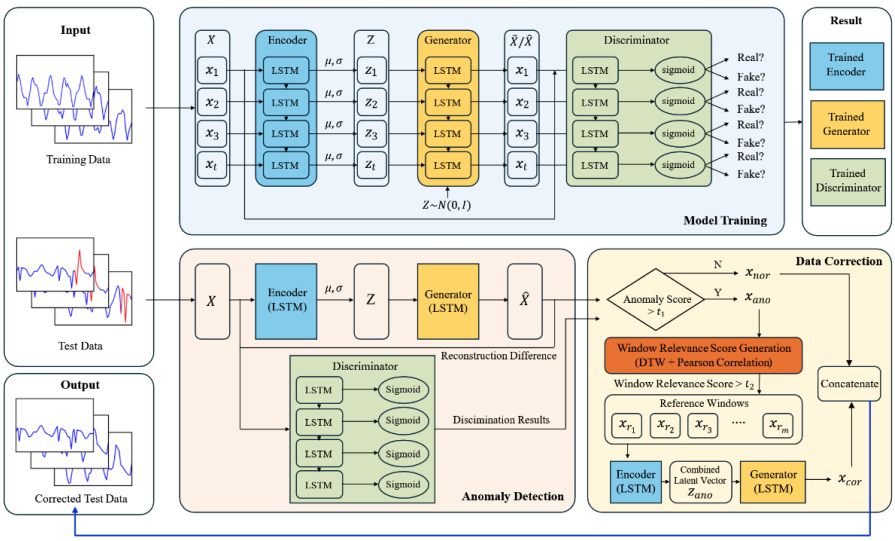

Fig. 5: Architecture of the Proposed Data Correction Method.

Top: model training of the LSTM-based VAE-GAN (encoder E outputs μ, σ, samples latent Z, generator G reconstructs $\hat{X}$, and discriminator D distinguishes Real vs. Fake), yielding trained E/G/D. Bottom-left: anomaly detection on test windows using the anomaly score from reconstruction difference $| X - \hat{X} |$ and discriminator output, producing anomalous windows x_{ano}. Bottom-right: data correction by computing DTW+Pearson window relevance scores, selecting reference windows $\{x_r\}$, concatenating/aggregating their latent vectors into z_{ano}, and generating corrected windows x_{cor} to replace x_{ano}, resulting in corrected test data.

Second, to correct the detected anomalous windows, we construct a window relevance matrix. This matrix is designed to identify the most relevant windows containing useful information for anomaly correction. The construction of the window relevance matrix incorporates calculations that account for both temporal characteristics and inter-variable relationships. Leveraging information from highly relevant windows, the generator of the trained VAE-GAN model regenerates windows that resemble normal values. By replacing the detected anomalous windows with these generated windows, we obtain the final corrected test data.

Figure 5 presents the detailed architecture of the proposed correction method. It consists of three main components: model training, where the VAE-GAN model is trained; anomaly detection, where the trained encoder, generator, and discriminator are used to identify anomalies; and data correction, where the detected anomalies are replaced with corrected values.

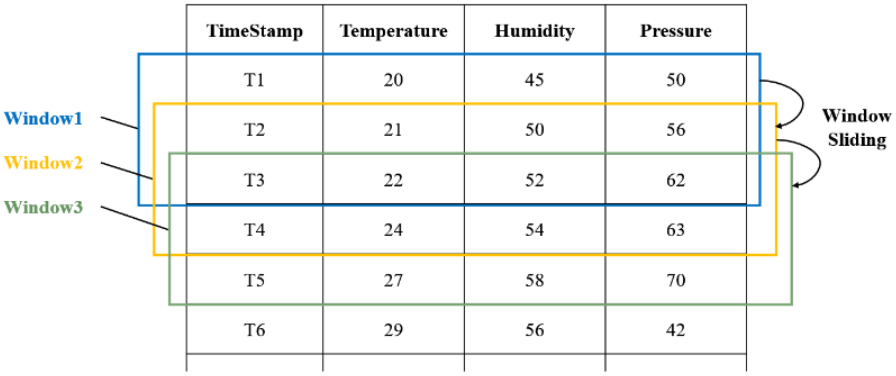

Fig. 6: Example of Applying the Sliding Window Technique.

3.1. Training LSTM-based VAE-GAN model

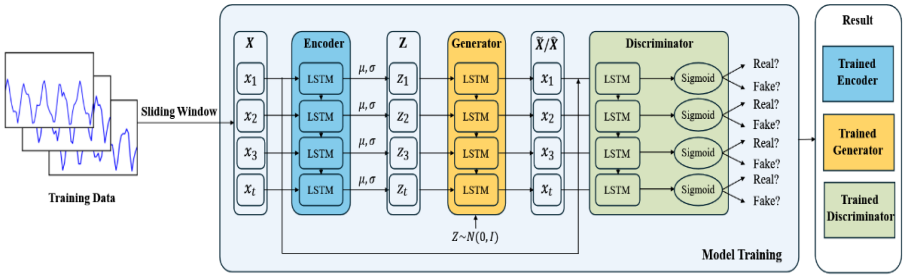

Fig. 7: Architecture of LSTM-based VAE-GAN Model.

Our LSTM-based VAE-GAN model leverages the complementary strengths of Variational Autoencoders (VAE) and generative adversarial networks (GAN) to learn the underlying distribution of normal patterns, enabling both anomaly detection and reconstruction. The architecture integrates LSTM networks to capture temporal dependencies, comprising three main components: an encoder, generator, and discriminator.

The data preprocessing phase employed a sliding window technique to prepare the input for the model. As illustrated in Figure 6, with a window size of 3 and a step size of 1, this technique generates sequential segments of the time series, preserving temporal relationships while creating suitable training instances. Given a multivariate time series dataset containing temperature, humidity, and pressure measurements across seven timestamps, we segmented the data into overlapping windows.

The LSTM-based encoder processes these windowed segments to generate latent space representations. By computing the mean and standard deviation of the encoded data, the encoder ensures the latent vectors conform to a normalized distribution, facilitating the subsequent generation process. This probabilistic encoding enables the model to capture the essential characteristics of normal operational patterns while maintaining the capability to generate realistic corrections for anomalous data points.

The generator accepts the latent vector encoded by the encoder as input and performs two functions: reconstructing the original data and generating novel samples by sampling from a Gaussian distribution. During training, the generator is optimized to produce both reconstructed and synthetic data that are sufficiently realistic to deceive the discriminator. Conversely, the discriminator is trained to accurately distinguish between authentic data and artificially generated or reconstructed samples, effectively classifying them as either real or fake. The architectural framework of our LSTM-based VAE-GAN model is illustrated in Figure 7. To ensure robust anomaly detection and correction capabilities, the model was trained exclusively on clean data windows containing no anomalous patterns.

3.2. Anomaly detection

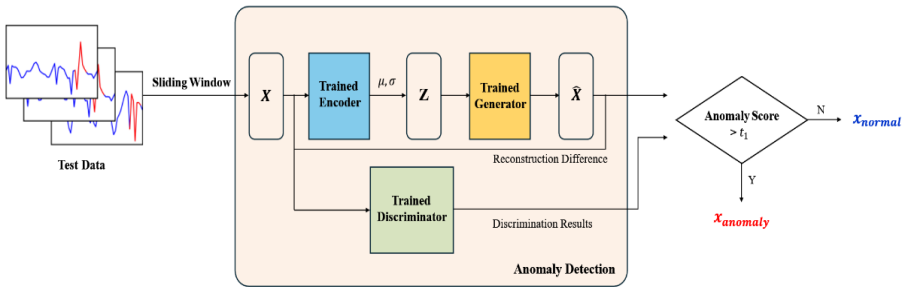

Fig. 8: Anomaly Detection Process.

Anomaly detection is conducted on test data containing anomalies using the trained model. Figure 8 illustrates the anomaly detection process with the trained VAE-GAN model. In this process, the anomaly score is computed by combining the reconstruction difference between the synthetic window ($\hat{X}$) generated by the trained generator and the real window (X), along with the discrimination results

$\mathrm{Dis}(\mathrm{X_{test}})$ from the trained discriminator. The anomaly score is defined as shown in Eqn. (2), and if the score exceeds a predefined threshold ($\mathrm{t_1}$), the corresponding window is identified as an anomaly ($\mathrm{x_{anomaly}}$).

$$\text{Anomaly Score} = (1-\alpha)\cdot\left|\mathrm{X_{test}} - \hat{\mathrm{X}}_{\mathrm{test}}\right| + \alpha\cdot\mathrm{Dis}(\mathrm{X_{test}}) \quad (2)$$

For example, if a given window contains anomalies, the reconstruction error —representing the difference between the generated and real windows —along with the discriminator's classification results, increases. Consequently, the anomaly score rises, leading to the detection of the window as anomalous.

3.3. Data correction

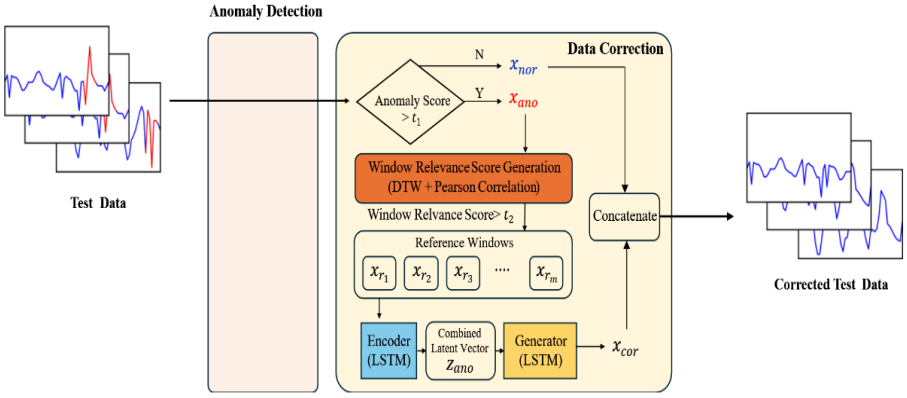

Fig. 9: Data Correction Process.

The LSTM-based VAE-GAN model is trained such that when a normal window is passed through the encoder to the generator, the generator produces a window resembling the normal window, while the discriminator differentiates between real and generated windows. In this study, we leverage the generator's ability to produce windows similar to normal ones to correct detected anomalous windows. Figure 9 illustrates the correction process using the trained encoder and generator. To incorporate additional window information during the correction process, a window relevance matrix is constructed. By utilizing information from highly relevant windows, the generator of the trained VAE-GAN model regenerates windows that closely resemble normal values. The final step in data correction involves replacing the detected anomalous windows with the generated normal windows.

3.3.1. Data correction process

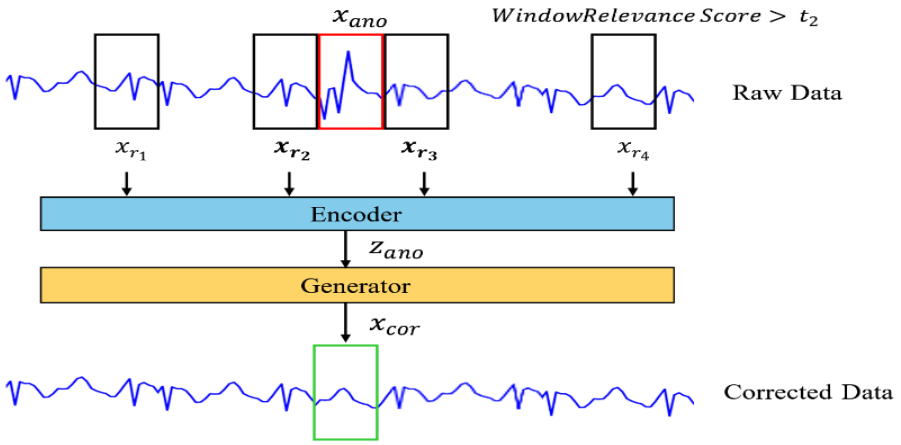

Fig. 10: Process of Correcting Anomalous Windows into Normal Windows.

To correct windows detected as anomalies and replace them with normal windows, we employ the trained encoder and generator. Figure 10 illustrates the process of correcting detected anomalous windows into normal ones. Given that an anomalous window ($\mathrm{x_{ano}}$) has been detected, the correction process proceeds as follows.

In the first step, it is necessary to identify windows that contain relevant information for replacing the anomalous window. Since windows with the closest similarity to the anomalous window are likely to share similar patterns, they are considered highly relevant. These windows ($\mathrm{x_{r_2}}, \mathrm{x_{r_3}}$) are defined as 'target windows', serving as reference points for identifying other highly relevant windows.

In the second step, all windows except for the detected anomalous ones are defined as 'search windows', and a window relevance matrix is constructed. Using this matrix, a window relevance score (see Eqn. (3)) is determined based on the similarity between windows, combining DTW similarity to capture temporal characteristics and Pearson correlation coefficients to account for inter-variable relationships. The details of constructing the window relevance matrix is discussed in Section 3.3.3.

In the third step, the most relevant windows are extracted by selecting those with a window relevance score above a predefined threshold ($\mathrm{t_2}$). These windows, including the most relevant target window, are defined as 'reference windows'. In the given example, when the windows $\mathrm{x_{r_1}}$ and $\mathrm{x_{r_4}}$ meet the threshold for high relevance, the reference windows $\mathrm{x_{r_1}}, \mathrm{x_{r_2}}\mathrm{x_{r_3}}$ and $\mathrm{x_{r_4}}$ are selected accordingly.

In the fourth step, the reference windows are used as input for the trained encoder. The latent vectors obtained from the encoder are then combined to form a combined latent vector (z_{ano}). By integrating information from multiple windows, this approach captures diverse patterns within the time series data, thereby improving the quality of the correction.

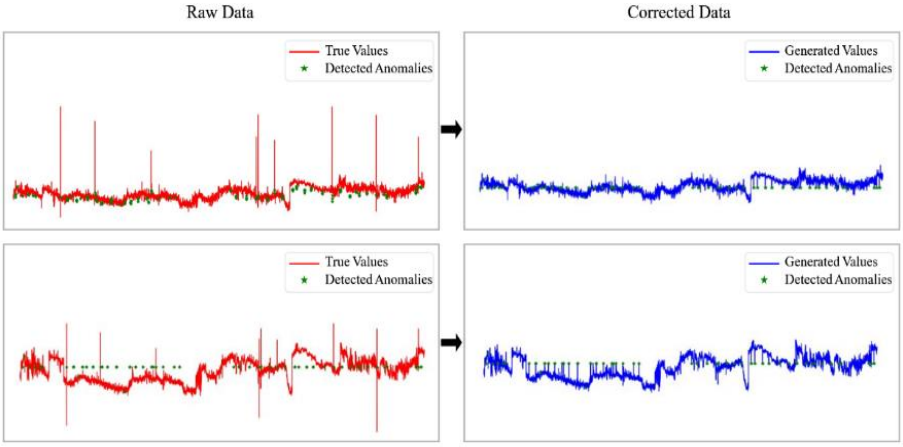

Fig. 11: Correction Results Using the Proposed Method.

Finally, the combined latent vector is fed into the trained generator to produce new normal values (x_{cor}). By replacing the anomalous windows with these generated values, the corrected data ultimately follows a normal pattern. Figure 11 visualizes the correction results for the HAI dataset from the experiment. The left side displays the raw data before correction, while the right side presents the corrected time series data obtained using the proposed correction method.

3.3.2. Correction methods based on anomaly types

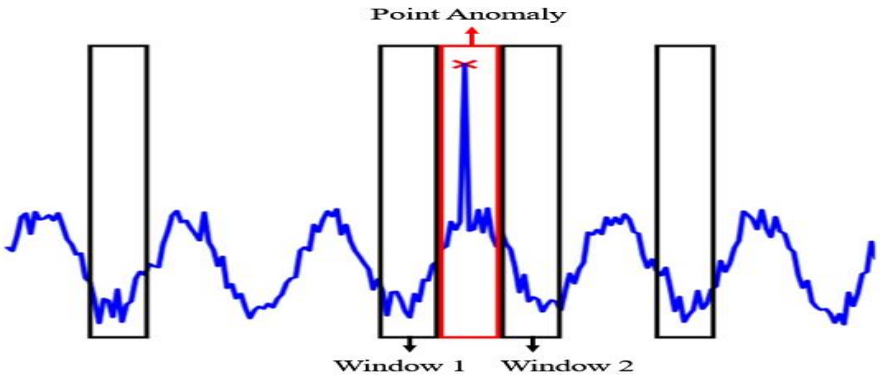

Fig. 12: Correction Process for Point Anomalies.

As previously mentioned, this study distinguishes between point anomalies and collective anomalies for correction, considering the characteristics of time series data. A point anomaly is a temporary deviation in which an individual value significantly differs from the surrounding data. Therefore, point anomaly correction is performed at the window level, utilizing surrounding windows as target windows to identify reference windows. Figure 12 illustrates the correction process for point anomalies. If a red-colored window is detected as an anomalous window, windows 1 and 2 are designated as target windows to construct the window relevance matrix. The correction is then performed using information from windows with a relevance score exceeding a predefined threshold.

The second method focuses on correcting collective anomalies. A collective anomaly occurs when multiple consecutive data points form an abnormal pattern. Consequently, reference windows are identified at the window group level, where a window group is defined as a set of consecutive windows, some of which contain a collective anomaly, corresponding to the number of consecutive anomalous windows. Since the search windows may overlap with other anomalous windows, point anomaly correction is performed first, followed by collective anomaly correction. Additionally, subsequent target windows may contain other anomalous windows. Therefore, for collective anomalies, only the preceding windows of the detected anomalous window are selected as target windows. Figure 13 illustrates the correction process for collective anomalies.

For an anomalous window, a window relevance matrix is constructed using window group 1 as the reference. The correction is then performed using the information from windows with a relevance score exceeding the predefined threshold.

3.3.3. Window relevance matrix

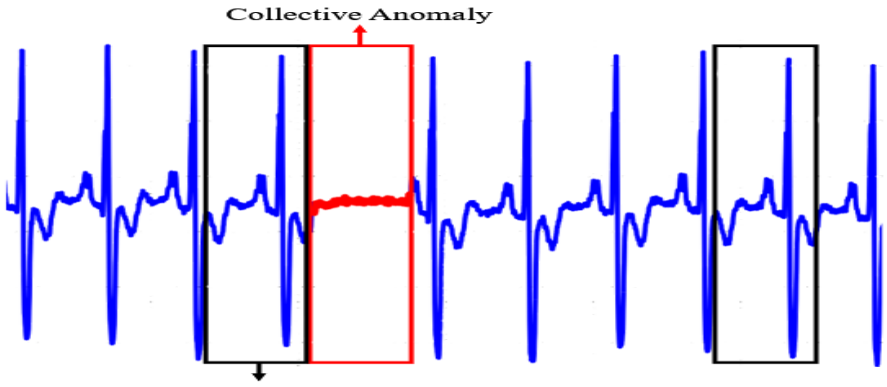

Fig. 13: Correction Process for Collective Anomalies.

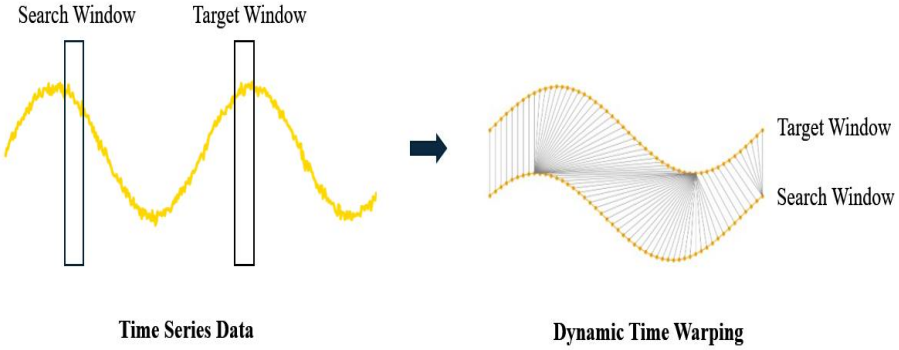

Fig. 14: Example of DTW Alignment Process for a Single Variable.

In this study, a window relevance matrix is constructed to identify windows that are most relevant to the target window. To achieve this, window relevance is computed using DTW and Pearson correlation coefficients to measure the similarity between two windows.

First, Dynamic Time Warping (DTW) is an algorithm that measures temporal pattern similarity between time series data by allowing temporal distortions, enabling effective similarity assessment even when data vary at different speeds. Figure 14 visually illustrates an example of DTW alignment between a target window and a search window. As shown in the figure, DTW is computed for each variable, and the final DTW value is obtained by averaging these individual DTW scores. This approach enables the identification of time series patterns beyond simple value comparisons. A smaller DTW value indicates a higher similarity in patterns.

Second, the Pearson correlation coefficient quantifies the linear relationship between variables and ranges from -1 and 1. Figure 15 illustrates an example of generating a correlation coefficient matrix for the target window (w_T) and search window (w_s). The difference between the two generated matrices is used to compare the similarity of inter-variable relationships between the two windows. In this study, we use the Frobenius norm to compute the distance $\|w_T - w_s\|_F$ between the two matrices, reflecting their similarity. A smaller distance indicates a higher similarity in correlation structures between the two windows.

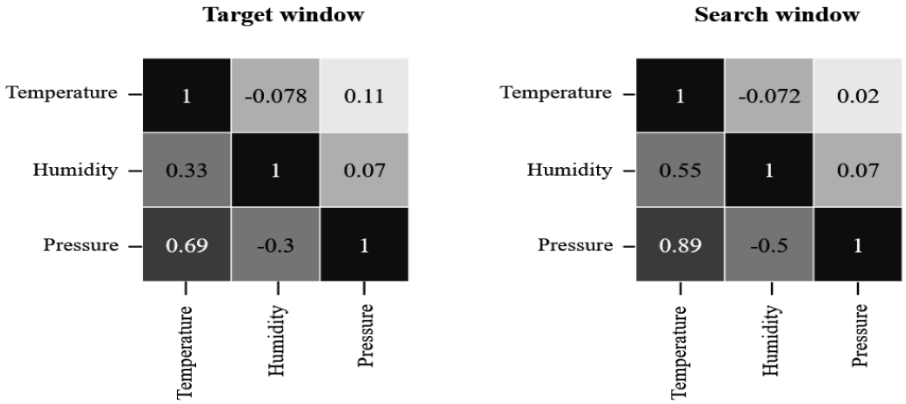

Fig. 15: Example of Pearson Correlation Matrices.

Consequently, the window relevance score is defined by combining DTW and Pearson correlation coefficients using the following formula:

$$\text{Window Relevance Score}(w_T, w_S) = \alpha \cdot \left(1 - N_{DTW}(w_T, w_S)\right) + \beta \cdot \left(1 - N_{Pearson}(w_T, w_S)\right) \quad (3)$$

Eqn. (3) calculates the degree of window relevance between a target window w_T and a search window w_S. $N_{DTW}(w_T, w_S)$ is the min-max normalized similarity score between w_T and w_S based on the DTW algorithm, which captures temporal similarity. Similarly, $N_{Pearson}(w_T, w_S)$ is the min-max normalized Pearson correlation coefficient between w_T and w_S, reflecting their inter-variable relationships. The final score is then computed by subtracting the normalized value from 1, ensuring that higher scores are assigned to more similar

windows. Additionally, weighting factors α and β are introduced to adjust the relative importance of temporal similarity and inter-variable correlation similarity. The final window relevance score ranges from 0 and 1, where values closer to 1 indicate higher relevance between the two windows. In this study, α=0.7 and β=0.3 were used, which will be empirically justified in Section 4.3.

4. Results

This study evaluates the effectiveness of the proposed correction method by indirectly assessing its impact on multivariate time series prediction models. The performance of these models before and after anomaly correction is compared to validate the effectiveness of the correction process.

4.1. Data setup

To evaluate the efficacy of the proposed method, we conducted experiments using publicly available datasets, including SWaT and WADI from SUTD's iTrust Lab, HAI from the National Security Research Institute, and SKAB from Kaggle.com.

1) SWaT (Secure Water Treatment): A water-treatment security/anomaly-detection dataset collected from a scaled industrial testbed. It contains normal operation data and cyberattack scenarios (e.g., sensor tampering, pump malfunctions), with sensor readings such as flow rate, pressure, and water level.
2) WADI (Water Distribution): A water-distribution dataset collected from a testbed emulating a large-scale network. It includes normal operation and attack scenarios (e.g., pipe leaks, data tampering, sensor disturbances), mainly providing flow-rate and pressure measurements.
3) HAI (HIL-based Augmented ICS): An industrial control system (ICS) security/anomaly-detection dataset from a testbed simulating processes such as steam-turbine generation and hydroelectric storage. It provides industrial sensor data (e.g., pressure, flow rate, temperature) under various cyberattack scenarios.
4) SKAB (Skoltech Anomaly Benchmark): A benchmark dataset for evaluating anomaly detection in cyber-physical system settings. It covers sensor failures and equipment malfunctions, with variables such as voltage, flow rate, and temperature.

Table 2 summarizes key information about the experimental datasets. For SWaT, WADI, and HAI datasets, variables with minimal variation were removed to ensure a more meaningful analysis.

Table 2: Empirical Dataset Information

Feature	Dataset SWaT	WADI	HAI	SKAB
# Original variables	51	103	78	8
# Training data	496.800	1.048.571	921.603	10.897
# Test data	449.919	172.801	402.005	7.265
# Selected variables	14	27	22	8
Anomaly ratio (%)	11.98	5.99	2.23	34.74

4.2. Experimental design

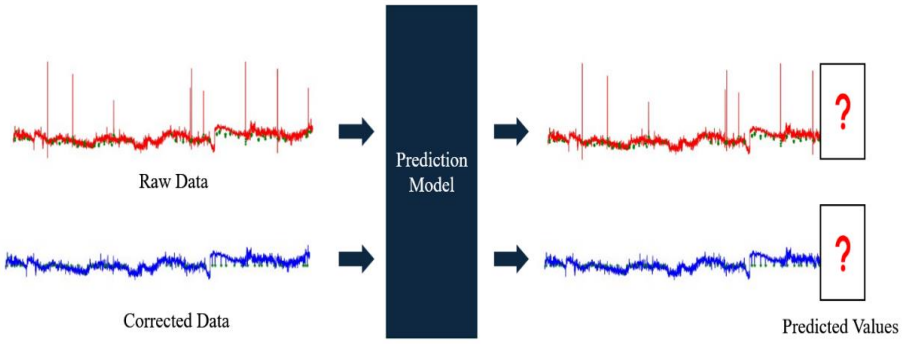

Fig. 16: Indirect Evaluation through Predictive Model Performance.

In multivariate time series data correction, the effectiveness of a correction method is typically evaluated indirectly by assessing the performance of a predictive model trained on the corrected data. Figure 16 illustrates the rationale behind using predictive model performance for indirect evaluation. A model trained in raw data containing anomalies is likely to experience decreased predictive accuracy. In contrast, a model trained on corrected data is expected to produce more stable and accurate predictions. Therefore, training with corrected data is anticipated to reduce prediction errors, ultimately demonstrating the effectiveness of the correction method.

In this study, the LSTM algorithm was used to build the predictive model. To quantitatively evaluate the predictive performance on both raw data with anomalies and corrected data, we use Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). A lower value of these metrics indicates better predictive model performance. Here, Y_i represents the actual values and $\hat{Y}_i$ represents the predicted values.

4.3. Performance evaluation

To compare the performance of the proposed correction method, predictive models were built using raw data, data with anomalies removed, and data corrected using existing correction methods. Additionally, we evaluated the effect of different threshold values (t_2) for the window relevance matrix when generating windows for data correction, comparing performance across different window sizes.

Table 3 presents the performance comparison of various correction methods using SWaT dataset with MAE and RMSE. The results indicate that the predictive model trained on data corrected by the proposed method achieved the best performance across all window sizes. Similarly, Table 4 presents the performance comparison using WADI dataset, confirming that the predictive model trained on corrected data from the proposed method outperformed other approaches, consistent with the SWaT results. Table 5 presents the performance comparison

for HAI dataset, and Table 6 shows the results for SKAB dataset, both of which further support the effectiveness of the proposed correction method. For each dataset, performance evaluation was conducted using window sizes of 30, 60, and 120. The best performance values for each window size are marked with an asterisk.

The results indicate that the proposed correction method consistently outperforms other methods across different window sizes (30, 60, and 120), achieving the lowest MAE and RMSE values. The predictive model trained on data corrected by the proposed method showed improvements ranging from 3.5% to 12.1% compared to raw data, 4.6% to 10.0% compared to data with anomalies removed, 0.3% to 15.1% compared to the Kalman filtering-based correction method, 3.3% to 8.7% compared to the mean-based correction method, 3.6% to 17.3% compared to the SVR-based correction method, and 0.1% to 18.4% compared to the LSTM-based correction method.

Table 3: Performance Comparison of Correction Methods on SWaT Dataset.

Window size	Data	MAE	RMSE
30	Raw Data	0.0466	0.0949
	Anomalies Removed	0.0473	0.0940
	Kalman Filtering	0.0443	0.0901
	Mean-Based Correction	0.0467	0.0921
	SVR-Based Correction	0.0447	0.0908
	LSTM-Based Correction	0.0450	0.0941
	Proposed Correction	*0.0414	*0.0854
60	Raw Data	0.0509	0.1011
	Anomalies Removed	0.0481	0.0938
	Kalman Filtering	0.0491	0.0931
	Mean-Based Correction	0.0647	0.0930
	SVR-Based Correction	0.0486	0.0930
	LSTM-Based Correction	0.0483	0.0920
	Proposed Correction	*0.0467	*0.0924
120	Raw Data	0.0451	0.0949
	Anomalies Removed	0.0471	0.0941
	Kalman Filtering	0.0445	0.0940
	Mean-Based Correction	0.0480	0.0931
	SVR-Based Correction	0.0466	*0.0901
	LSTM-Based Correction	0.0450	0.0941
	Proposed Correction	*0.0444	0.0908

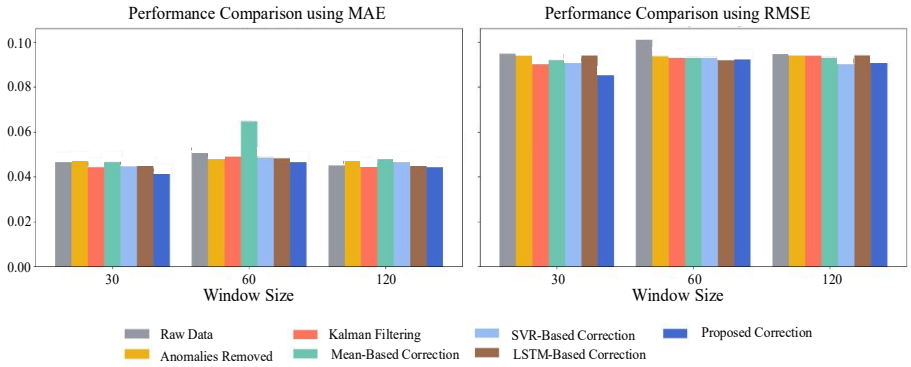

Fig. 17: Performance Comparison of Correction Methods for SWaT Dataset.

Table 4: Performance Comparison of Correction Methods on WADI Dataset.

Window size	Data	MAE	RMSE
30	Raw Data	0.0883	0.1442
	Anomalies Removed	0.0963	0.1544
	Kalman Filtering	0.0882	*0.1430
	Mean-Based Correction	0.0925	0.1515
	SVR-Based Correction	0.0949	0.1543
	LSTM-Based Correction	0.0881	0.1440
	Proposed Correction	*0.0880	0.1439
60	Raw Data	0.0927	0.1444
	Anomalies Removed	0.0966	0.1539
	Kalman Filtering	0.0864	0.1459
	Mean-Based Correction	0.0941	0.1529
	SVR-Based Correction	0.0924	0.1465
	LSTM-Based Correction	0.0920	0.1465
	Proposed Correction	*0.0865	*0.1418
120	Raw Data	0.0991	0.1564
	Anomalies Removed	0.0920	0.1563
	Kalman Filtering	0.0999	0.1583
	Mean-Based Correction	0.0946	0.1518
	SVR-Based Correction	0.0949	0.1549
	LSTM-Based Correction	0.0990	0.1530
	Proposed Correction	*0.0910	*0.1504

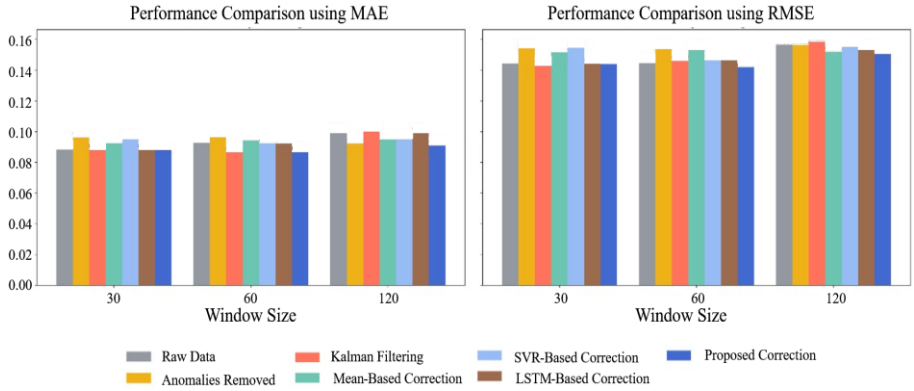

Fig. 18: Performance Comparison of Correction Methods for WADI Dataset.

Table 5: Performance Comparison of Correction Methods on HAI Dataset.

Window size	Data	MAE	RMSE
30	Raw Data	0.0510	0.0758
	Anomalies Removed	0.0508	0.0756
	Kalman Filtering	0.0501	0.0753
	Mean-Based Correction	0.0513	0.0758
	SVR-Based Correction	0.0531	0.0866
	LSTM-Based Correction	0.0501	0.0756
	Proposed Correction	*0.0433	*0.0755
60	Raw Data	0.0489	0.0740
	Anomalies Removed	0.0487	*0.0738
	Kalman Filtering	0.0473	0.0739
	Mean-Based Correction	0.0496	0.0739
	SVR-Based Correction	0.0519	0.0866
	LSTM-Based Correction	0.0483	0.0865
	Proposed Correction	*0.0432	0.0760
120	Raw Data	0.0570	0.0850
	Anomalies Removed	0.0574	0.0739
	Kalman Filtering	0.0398	0.0726
	Mean-Based Correction	0.0493	0.0739
	SVR-Based Correction	0.0516	0.0868
	LSTM-Based Correction	0.0515	0.0826
	Proposed Correction	*0.0387	*0.0706

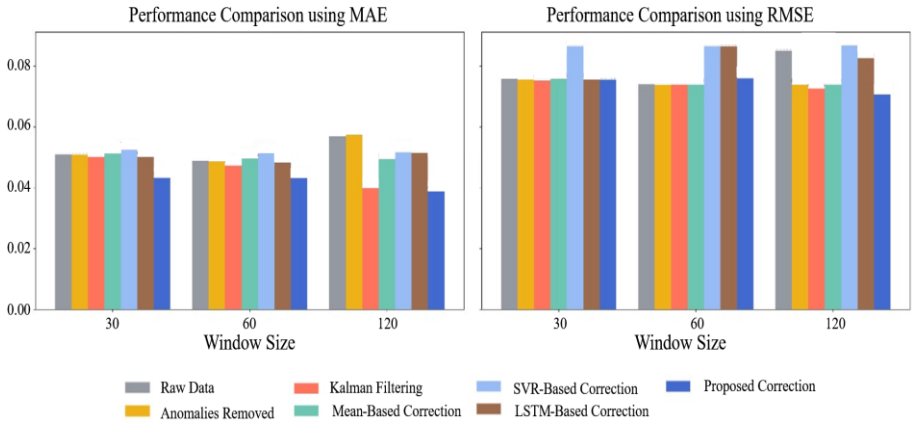

Fig. 19: Performance Comparison of Correction Methods for HAI Dataset.

Table 6: Performance Comparison of Correction Methods on SKAB Dataset.

Window size	Data	MAE	RMSE
30	Raw Data	0.1531	0.1758
	Anomalies Removed	0.1508	0.1756
	Kalman Filtering	0.1501	0.1756
	Mean-Based Correction	0.1513	0.1758
	SVR-Based Correction	0.1531	0.1866
	LSTM-Based Correction	0.1451	0.1757
	Proposed Correction	*0.1433	*0.1755
60	Raw Data	0.1489	0.1740
	Anomalies Removed	0.1487	0.1738
	Kalman Filtering	0.1481	0.1730
	Mean-Based Correction	0.1496	0.1739
	SVR-Based Correction	0.1519	0.1866
	LSTM-Based Correction	0.1478	0.1899
	Proposed Correction	*0.1432	*0.1726
120	Raw Data	0.1570	0.1850
	Anomalies Removed	0.1574	0.1739

Kalman Filtering	0.1573	0.1861
Mean-Based Correction	0.1493	0.1739
SVR-Based Correction	0.1516	0.1868
LSTM-Based Correction	0.1432	0.1738
Proposed Correction	*0.1387	*0.1706

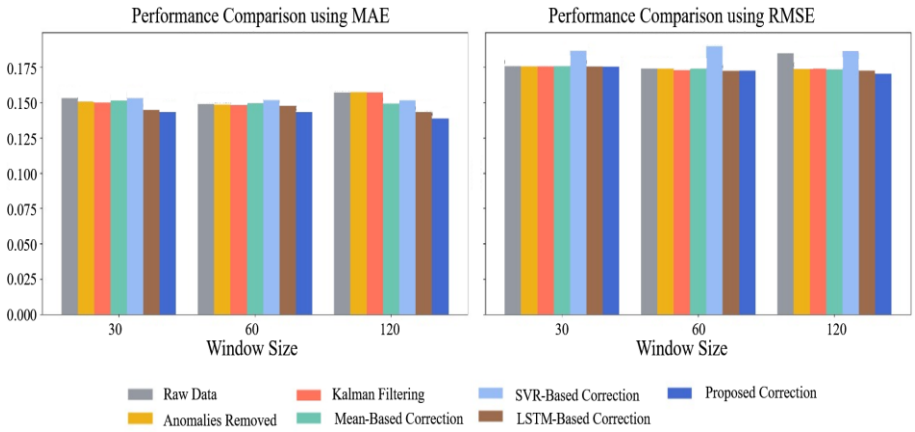

Fig. 20: Performance Comparison of Correction Methods for SKAB Dataset.

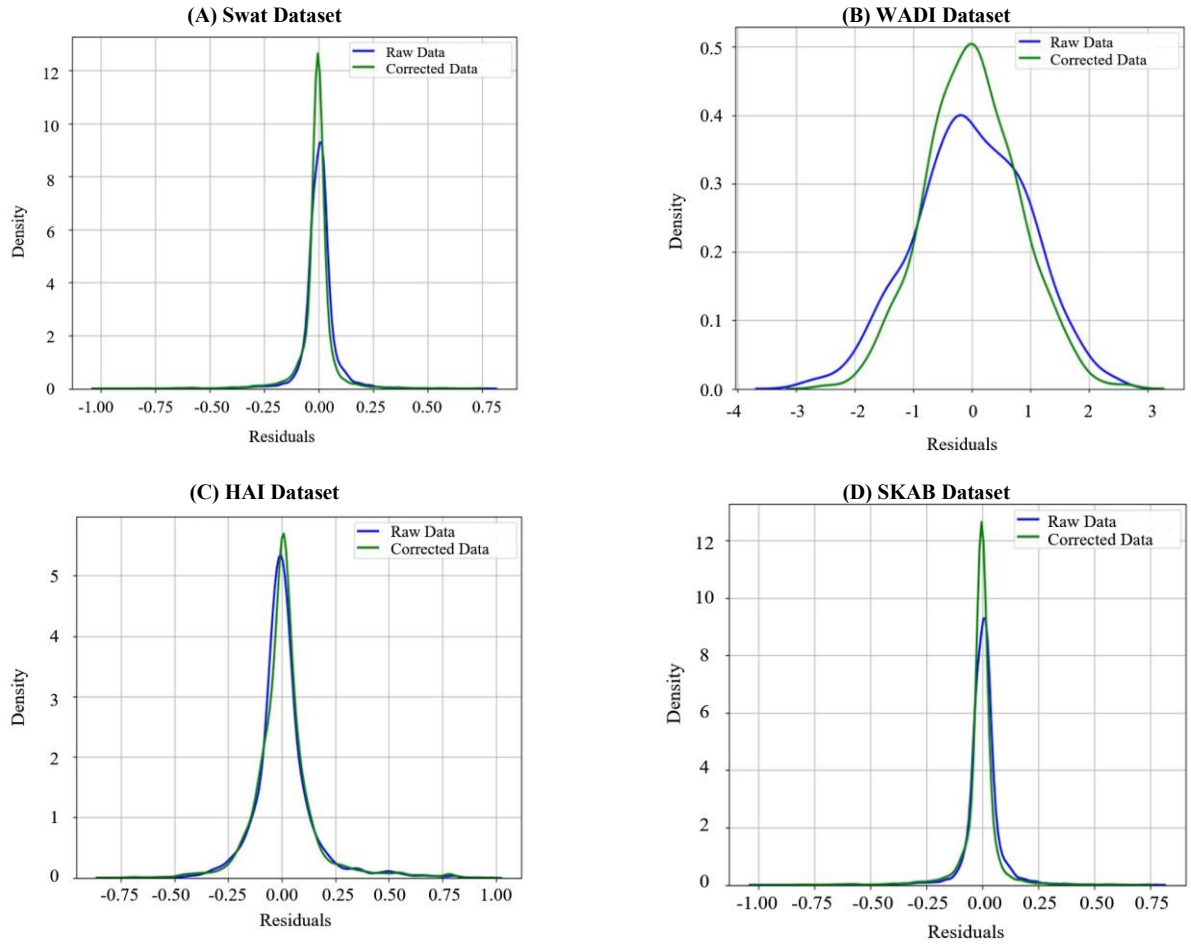

Fig. 21: Residual Distribution Comparison of Predictive Models.

Moreover, it is preferable to compare the residuals between predicted values and actual values for each dataset after applying the predictive model. Figure 21 visualizes the residual distribution of the raw data and corrected data using kernel density estimation (KDE) curves, providing a visual representation of how concentrated the residuals are around zero; that is, x-axis (Residuals) represents the difference between the predicted values and actual values. y-axis (Density) indicates the frequency of residual values, showing how concentrated they are around zero. Therefore, if the residuals in the corrected data are more densely concentrated around zero, it indicates that the corrected data improves prediction accuracy compared to the raw data (blue line). This suggests that the predictive model trained on data by the proposed method results in smaller residuals compared to the model trained on raw data, further confirming improved prediction accuracy. The residual distribution comparisons across the four datasets consistently demonstrate the effectiveness of the proposed correction method in improving prediction accuracy. In all cases, the corrected data (green line) exhibits a higher peak around zero and a narrower spread compared to the raw data (blue line), indicating a significant reduction in prediction errors. Additionally, the corrected data distributions show lower variance, suggesting that the correction method successfully minimizes the impact of anomalies, making the predictive models more stable. While the degree of improvement varies across datasets, the general trend remains consistent: corrected data results in smaller

residuals, and predictive models trained on corrected data achieve better overall performance. The results from different datasets further confirm the versatility of the proposed method in various domains, including industrial systems, cybersecurity, and water treatment.
These findings validate that the proposed correction method effectively enhances data quality and predictive model performance by addressing anomalies while preserving the underlying temporal relationships in multivariate time series data. The concentration of residuals around zero and the reduction in variance indicate that the correction method significantly improves the reliability of deep learning models. This consistent improvement across multiple datasets highlights the robustness of the approach, demonstrating its potential for broad applicability in anomaly detection and data imputation tasks.

Table 7: Sensitivity of Window Relevance Weighting for SWaT Dataset.

α (DTW weight)	β (Correlation weight)	MAE	RMSE
0.3	0.7	0.0498	0.0938
0.5	0.5	0.0455	0.0912
0.7	0.3	*0.0414	*0.0854
0.9	0.1	0.0424	0.0883

In addition, as for Eqn. (3), we conducted a sensitivity study on the weighting parameters in the window relevance score for SWaT dataset, as shown in Table 7. While α controls temporal alignment via DTW and β captures inter-variable dependency via correlation, we found that α=0.7 and β=0.3 provided the most stable improvement in predictive accuracy, indicating that temporal alignment plays a slightly more dominant role in identifying correction-relevant windows for the evaluated datasets.

4.4. Computational complexity and scalability

We analyze the computational cost of the proposed framework, which consists of windowing, (offline) training of the LSTM-based VAE-GAN, (online) anomaly detection, and selective correction using a correlation-aware window relevance matrix computed from DTW and Pearson correlation. Let d be the number of variables, T the total number of time steps, L the window length, and s the stride. The number of windows is $N \approx \left\lfloor \frac{T-L}{s} \right\rfloor + 1$. Let K be the number of target windows used to build the relevance matrix and M the number of candidate search windows (typically $M \leq N$ after excluding detected anomalous windows). Let R denote the number of selected relevant windows used for correction (windows whose relevance scores exceed a given threshold).

1) Windowing cost: Constructing sliding windows is $O(NLd)$ time to read/arrange the data (or $O(Td)$ in a streaming implementation with a ring buffer) and $O(Ld)$ working memory for real-time processing.
2) VAE-GAN (offline) training: The LSTM-based encoder and generator/discriminator are trained on clean windows only. Let h be the LSTM hidden size and E the number of epochs. The dominant cost per window forward/backward pass is approximately proportional to the LSTM operations, i.e., $O(L \cdot (dh + h^2))$ per module. Therefore, the offline training cost is $O(E \cdot N_{train} \cdot L \cdot (dh + h^2))$ where N_{train} is the number of clean training windows. This cost is paid once and does not affect online deployment except when periodic retraining is required.
3) Anomaly detection (online inference): For each incoming window, anomaly scoring requires a forward pass through the encoder/generator/discriminator to compute reconstruction difference and discriminator output. The per-window detection cost is $O(L \cdot (dh + h^2))$, which is linear in window length and can be parallelized via batching across windows on GPUs.
4) Window relevance matrix computation: To correct an anomalous window, we compute window relevance scores by combining DTW similarity and Pearson correlation between target windows and search windows. DTW is computed per variable and averaged across variables. The classic DTW dynamic program is $O(L^2)$per variable, hence $O(dL^2)$ per window pair. Pearson correlation per variable between two length-L windows is $O(L)$; thus $O(dL)$ per window pair (or $O(L)$ if aggregated carefully with precomputed sums, but still linear in L). For K target windows and M search windows, a straightforward relevance matrix costs: $O(KM \cdot (dL^2 + dL)) \approx O(KMdL^2)$. In practice, DTW dominates for large L, making relevance computation the primary bottleneck.
5) Correction cost using selected relevant windows: After selecting R relevant windows whose relevance scores exceed a threshold, correction encodes these windows to latent vectors, combines them, and generates replacement values using the trained generator. The correction stage cost is $O(R \cdot L \cdot (dh + h^2)) + O(L \cdot (dh + h^2))$, where the first term is encoding R windows and the second term is generating the corrected window. Typically, $R \ll M$, so this stage is not the dominant cost.

Although the proposed correction framework shows effectiveness on the evaluated datasets, applying it to substantially larger-scale data (e.g., higher sampling rates, longer monitoring horizons, more variables, and multi-site deployments) introduces several practical scalability challenges.
First, candidate-window growth amplifies relevance computation cost. Because the method computes a window relevance score by combining DTW-based temporal alignment and Pearson-correlation-based dependency assessment (weighted by parameters such as α and β), the number of candidate windows increases with the data horizon, and the total number of window-pair evaluations can grow rapidly. In particular, DTW can dominate runtime for longer windows, making relevance computation the primary bottleneck when the monitoring period becomes large or when the window length is increased. Second, memory and I/O overhead can become non-trivial. Maintaining a large pool of raw candidate windows (or materializing a large relevance matrix) can impose substantial memory and storage costs and may degrade throughput due to frequent disk access in long-horizon settings. Third, anomaly bursts can increase end-to-end latency. In operational environments, anomalies may occur in bursts or across multiple channels simultaneously; repeated "retrieve relevant windows → correct" cycles can accumulate and increase latency, potentially violating near real-time requirements. Finally, distribution shifts and heterogeneity become more prominent at scale. Compared to curated benchmark datasets, real-world large-scale deployments often exhibit concept drift across seasons, operating conditions, and sites, as well as heterogeneous sensor configurations and missingness patterns, which may degrade both detection thresholds and correction fidelity if the model and parameters remain fixed.
To mitigate these issues, we propose several practical strategies for large-scale deployment.

1) Two-stage retrieval to reduce DTW evaluations: instead of computing DTW against all candidate windows, we first shortlist candidates using a low-cost proxy (e.g., distance in the encoder's latent space, z-normalized L_2, or a fast correlation pre-filter), and then apply DTW-based relevance only to the top M'candidates ($M' \ll M$).
2) Constrained/approximate DTW and pruning: employing band-constrained DTW, early abandoning, or lower-bound pruning reduces DTW cost while preserving high-quality matches.
3) Compact caching and indexing: storing compact representations (latent vectors or summary statistics) rather than raw windows for long-term candidate pools reduces memory I/O and enables efficient approximate nearest-neighbor search.

4) Batching and parallelization: both relevance evaluation across window pairs and per-variable computations are highly parallelizable and can be accelerated via GPU batching or multi-core processing.
5) Drift-aware operation: periodic re-training and adaptive thresholding (for anomaly scoring and relevance selection) improve robustness under concept drift, and domain-aware modularization (e.g., grouping similar sites/sensors) stabilizes performance under heterogeneity.

Collectively, these strategies bound the number of expensive DTW computations, control memory growth, and preserve throughput, enabling the proposed framework to scale beyond the experimental data regime toward large-scale streaming deployments. The online per-window detection cost is linear in L, whereas the naïve relevance computation is quadratic in L due to DTW. Therefore, the overall scalability is primarily governed by how aggressively we reduce the number of DTW evaluations and/or constrain DTW computation, enabling feasible deployment in streaming sensor environments.

5. Conclusion

This study proposed a correction method using an LSTM-based VAE-GAN anomaly detection model to improve the quality of multivariate time series data. Unlike conventional correction methods, the proposed approach enables simultaneous anomaly detection and correction within a single training process. To demonstrate its effectiveness, the performance of predictive models trained on data corrected by the proposed method was compared indirectly. The results confirmed that the predictive model trained on quality-enhanced data from the proposed correction method outperformed models trained on data corrected using existing approaches.

Furthermore, the residual distribution comparisons across multiple datasets (SWaT, WADI, HAI, and SKAB) provided additional evidence of the method's effectiveness. The corrected data consistently exhibited narrower residual distributions, with residuals more densely concentrated around zero, indicating reduced prediction errors. This trend was observed across all datasets, highlighting the robustness and versatility of the proposed method in improving predictive accuracy. The results also confirmed that the proposed method outperforms the simple removal of anomalies, emphasizing the importance of correcting anomalies rather than discarding them. These findings demonstrate that incorporating the underlying characteristics of multivariate time series data into the correction process enhances data quality and contributes to the development of more reliable AI models.

In this study, the VAE-GAN model was trained using benchmark datasets with labeled anomalies, ensuring that the model was trained exclusively on normal data. However, in real-world scenarios, data often contains anomalies without explicit labels. The proposed method is trained on clean (non-anomalous) windows, and its detection/correction performance may degrade when truly clean data are scarce or when anomaly labels are unavailable in real-world deployments. In addition, training the LSTM-based VAE-GAN incurs a non-trivial offline computational cost, and re-training may be required under concept drift caused by changes in operating conditions. Finally, the DTW-based component used in the window relevance computation can become a computational bottleneck for long windows, high sampling rates, or large candidate pools, which may affect scalability if not carefully optimized. In online monitoring, anomaly scoring is performed window-wise and can be efficiently executed via batching and parallelization. The main factor affecting real-time feasibility is the window retrieval step, where DTW-based relevance evaluation across a large candidate set may increase latency.

Therefore, future research will focus on developing a correction method that enhances anomaly detection and correction performance in unlabeled real-world datasets. Additionally, further investigation into adaptive correction strategies that can dynamically adjust to different types of anomalies and domain-specific constraints would enhance the applicability of the proposed method.

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2022R1A2C1011937).

References

[1] Ge, M., Bangui, H., & Buhnova, B. (2018). Big data for internet of things: a survey. *Future generation computer systems*, 87, 601-614. https://doi.org/10.1016/j.future.2018.04.053.

[2] Choi, J., & Shin, Y. (2021). Design of Efficient Big Data Collection Method based on Mass IoT devices. *The Journal of Korea Institute of Information, Electronics, and Communication Technology*, 14(4), 300-306.

[3] Ren, S., Kim, J. S., Cho, W. S., Soeng, S., Kong, S., & Lee, K. H. (2021, April). Big data platform for intelligence industrial IoT sensor monitoring system based on edge computing and AI. In 2021 International conference on artificial intelligence in information and communication (*ICAIIC*) (pp. 480-482). IEEE. https://doi.org/10.1109/ICAIIC51459.2021.9415189.

[4] Farahani, M. A., El Kalach, F., Harper, A., Mccormick, M. R., Harik, R., & Wuest, T. (2025). Time-series forecasting in smart manufacturing systems: An experimental evaluation of the state-of-the-art algorithms. *Robotics and Computer-Integrated Manufacturing*, 95, 103010. https://doi.org/10.1016/j.rcim.2025.103010.

[5] Su, N., Huang, S., & Su, C. (2024). Elevating smart manufacturing with a unified predictive maintenance platform: The synergy between data warehousing, Apache spark, and machine learning. *Sensors*, 24(13), 4237. https://doi.org/10.3390/s24134237.

[6] Zamanzadeh Darban, Z., Webb, G. I., Pan, S., Aggarwal, C., & Salehi, M. (2024). Deep learning for time series anomaly detection: A survey. *ACM Computing Surveys*, 57(1), 1-42. https://dl.acm.org/doi/10.1145/3691338.

[7] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys* (*CSUR*), 41(3), 1-58. https://doi.org/10.1145/1541880.1541882.

[8] Wang, F., Jiang, Y., Zhang, R., Wei, A., Xie, J., & Pang, X. (2025). A survey of deep anomaly detection in multivariate time series: taxonomy, applications, and directions. *Sensors*, 25(1), 190. https://doi.org/10.3390/s25010190.

[9] Seba, A. M., Gemeda, K. A., & Ramulu, P. J. (2024). Prediction and classification of IoT sensor faults using hybrid deep learning model. *Discover Applied Sciences*, 6(1), 9. https://doi.org/10.1007/s42452-024-053633-7.

[10] Garg, A., Zhang, W., Samaran, J., Savitha, R., & Foo, C. S. (2021). An evaluation of anomaly detection and diagnosis in multivariate time series. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6), 2508-2517. https://doi.org/10.1109/TNNLS.2021.3105827..

[11] Niu, Z., Yu, K., & Wu, X. (2020). LSTM-based VAE-GAN for time-series anomaly detection. *Sensors*, 20(13), 3738. https://doi.org/10.3390/s20133738.

[12] Mathur, A. P., & Tippenhauer, N. O. (2016, April). SWaT: A water treatment testbed for research and training on ICS security. In *2016 international workshop on cyber-physical systems for smart water networks* (*CySWater*) (pp. 31-36). IEEE. https://doi.org/10.1109/CySWater.2016.7469060.

[13] Ahmed, C. M., Palleti, V. R., & Mathur, A. P. (2017, April). WADI: a water distribution testbed for research in the design of secure cyber physical systems. In *Proceedings of the 3rd international workshop on cyber-physical systems for smart water networks* (pp. 25-28). https://dl.acm.org/doi/10.1145/3055366.3055375.

[14] Shin, H. K., Lee, W., Yun, J. H., & Min, B. G. (2021, August). Two ICS security datasets and anomaly detection contest on the HIL-based augmented ICS testbed. In *Proceedings of the 14th Cyber Security Experimentation and Test Workshop* (pp. 36-40). https://dl.acm.org/doi/10.1145/3474718.3474719.

[15] Fisch, A. T., Eckley, I. A., & Fearnhead, P. (2022). Subset multivariate collective and point anomaly detection. *Journal of Computational and Graphical Statistics*, 31(2), 574-585. https://doi.org/10.1080/10618600.2021.1987257.

[16] Zhanwei, S., & Zenghui, L. (2019). Abnormal detection method of industrial control system based on behavior model. *Computers & Security*, 84, 166-178. https://doi.org/10.1016/j.cose.2019.03.009

[17] Wang, H., Bah, M. J., & Hammad, M. (2019). Progress in outlier detection techniques: A survey. *IEEE Access*, *7*, 107964-108000. https://doi.org/10.1109/ACCESS.2019.2932762.

[18] Chalapathy, R., & Chawla, S. (2019). Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*.

[19] Boukerche, A., Zheng, L., & Alfandi, O. (2020). Outlier detection: Methods, models, and classification. *ACM Computing Surveys (CSUR)*, *53*(3), 1-37. https://dl.acm.org/doi/10.1145/3381028.

[20] Garg, A., Zhang, W., Samaran, J., Savitha, R., & Foo, C. S. (2021). An evaluation of anomaly detection and diagnosis in multivariate time series. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6), 2508-2517. https://doi.org/10.1109/TNNLS.2021.3105827.

[21] Ergen, T., & Kozat, S. S. (2019). Unsupervised anomaly detection with LSTM neural networks. *IEEE transactions on neural networks and learning systems*, 31(8), 3127-3141. https://doi.org/10.1109/TNNLS.2019.2935975.

[22] Chen, Z., Yeo, C. K., Lee, B. S., & Lau, C. T. (2018, April). Autoencoder-based network anomaly detection. In *2018 Wireless telecommunications symposium* (*WTS*) (pp. 1-5). IEEE. https://doi.org/10.1109/WTS.2018.8363930.

[23] Xia, X., Pan, X., Li, N., He, X., Ma, L., Zhang, X., & Ding, N. (2022). GAN-based anomaly detection: A review. *Neurocomputing*, 493, 497-535. https://doi.org/10.1016/j.neucom.2021.12.093

[24] Wang, X., & Wang, C. (2019). Time series data cleaning: A survey. *IEEE Access*, *8*, 1866-1881. https://doi.org/10.1109/ACCESS.2019.2962152.

[25] Lee, M. K., Moon, S. H., Kim, Y. H., & Moon, B. R. (2014, October). Correcting abnormalities in meteorological data by machine learning. In *2014 IEEE International Conference on Systems, Man, and Cybernetics* (*SMC*) (pp. 888-893). IEEE. https://doi.org/ 10.1109/SMC.2014.6974024.

[26] Baghoussi, Y., Soares, C., & Mendes-Moreira, J. (2024). Corrector LSTM: Built-in training data correction for improved time-series forecasting. *Neural Computing and Applications*, 36(26), 16213-16231. https://doi.org/10.1007/s00521-024-09962-x.

[27] Zhang, J., & Yin, P. (2019, November). Multivariate time series missing data imputation using recurrent denoising autoencoder. In *2019 IEEE International Conference on Bioinformatics and Biomedicine* (*BIBM*) (pp. 760-764). IEEE. https://doi.org/10.1109/BIBM47256.2019.8982996.

[28] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63(11), 139-144. https://dl.acm.org/doi/10.1145/3422622.