Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.084688

ARTICLE

A Multi-Scale Time-Series Anomaly Detection Approach for Modeling the Time-Lagged Effects of Exogenous Variables

Yuanzhao Shang[1], **Xin Liu**[1,2,*] **and Fengbiao Zan**[1]

[1]School of Intelligence Science and Engineering, Qinghai Minzu University, Xining, China
[2]Qinghai Provincial Key Laboratory of IoT, Xining, China

*Corresponding Author: Xin Liu. Email: lx@qhmu.edu.cn

Received: 27 April 2026; Accepted: 26 June 2026; Published: 13 August 2026

ABSTRACT: Multivariate time-series anomaly detection is widely used to identify abnormal operating patterns in complex monitoring systems. Exogenous or contextual inputs can provide useful information for anomaly detection, but their effects on endogenous variables may appear with temporal delays. Direct synchronous modeling is therefore insufficient for capturing delayed response patterns. This study proposes EMS-uDTWAD, a multi-scale anomaly detection framework that combines explicit lag alignment with uncertainty-aware normal-pattern matching. First, the time-series data are divided into fine-grained and coarse-grained windows. At the coarse-grained scale, a lag alignment module estimates delayed responses from exogenous or contextual inputs to endogenous variables and constructs lag-enhanced joint windows. These lag-enhanced windows are then matched against a normal reference library using uncertainty-aware dynamic time warping (uDTW). Thus, uDTW is used for global normal-pattern matching rather than for directly estimating inter-channel time lags. At the fine-grained scale, a graph embedding module models local structural dependencies and cross-channel correlations through a learned adjacency representation. Finally, the anomaly scores from the two branches are fused to obtain the final detection result. Experiments are conducted on two real-world datasets, MSDS and SWaT, and one synthetic Linear Dataset. MSDS is used as a proxy auxiliary-context setting, SWaT is used as a real industrial operational-context setting, and the Linear Dataset provides native exogenous variables. Across these settings, EMS-uDTWAD improves the F1 score over the strongest compared baseline by 2.0 percentage points on MSDS, 1.0 percentage point on the Linear Dataset, and 0.8 percentage points on SWaT. The ablation results further show that the lag-enhanced representation is more useful than direct synchronous concatenation when delayed contextual responses are present.

KEYWORDS: Time series anomaly detection; exogenous and contextual inputs; time-lag modeling; uncertainty-aware dynamic time warping

1 Introduction

Time-series anomaly detection is widely used in financial systems, geoscience, manufacturing, healthcare, and industrial monitoring [1–4]. In multivariate settings, the detector usually receives measurements from multiple sensors and identifies abnormal temporal changes or abnormal cross-variable relationships. Classical time-series similarity and outlier-detection methods remain useful in simple scenarios [5–8], but recent studies have increasingly adopted reconstruction-based [9,10], contrastive-learning-based [11,12], and forecasting-based models [13,14]. Recent work has also examined adversarial attack detection for deep learning-based soft sensors through bidirectional consistency discrimination [15] and decomposition-based

multi-scale Transformer architectures for time-series anomaly detection [16]. These studies improve robustness and multi-scale temporal representation, but they usually model the observed variables themselves and pay less attention to delayed responses caused by contextual or exogenous inputs.

In practical monitoring systems, some abnormal-looking changes in endogenous variables may be normal responses to external or operational conditions [17]. For example, in CNC power-consumption monitoring, ambient temperature or workload changes may affect power consumption after a short delay. Fig. 1 schematically illustrates this delayed-response relationship. If the model only uses synchronous concatenation, the delayed response may be incorrectly treated as an anomaly. Existing forecasting studies have introduced exogenous variables into Transformer-based models to improve prediction under heterogeneous inputs [18]. Sequence-alignment methods, including DTW-related formulations [19], provide another way to compare temporally shifted patterns. However, these two ideas are rarely combined in anomaly detection to distinguish true anomalies from normal delayed contextual responses.

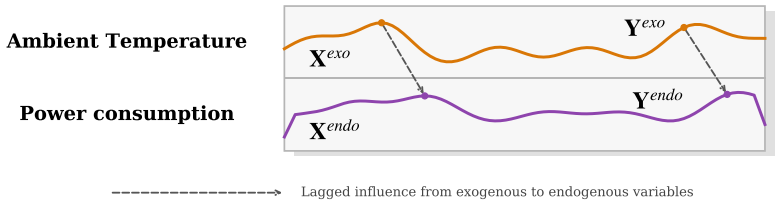

Figure 1: Schematic illustration of a delayed response from an exogenous change to an endogenous variable.

Two limitations are especially relevant to this study. First, most anomaly detectors focus on endogenous temporal dependencies and cross-variable correlations, while delayed contextual responses are often ignored or handled by direct concatenation. Second, DTW-based matching can compare temporally shifted sequences, but conventional DTW and sDTW do not explicitly use time-step-level uncertainty during matching. Noisy or locally distorted segments may therefore dominate the matching distance and affect the anomaly score.

To address these issues, this paper proposes EMS-uDTWAD, a multi-scale anomaly detection framework for time series with exogenous or auxiliary contextual inputs. The framework separates lag modeling from sequence matching. At the coarse-grained scale, a lag alignment mechanism estimates delayed exogenous-to-endogenous responses and constructs lag-enhanced joint windows, which are then matched with a normal reference library using uDTW. At the fine-grained scale, a graph embedding module captures local structural dependencies and cross-channel correlations. We do not claim that multi-scale windowing, Graph VAE, or uDTW is newly invented in isolation. The main novelty lies in organizing these components into a lag-aware anomaly detection framework, where explicit lag estimation, uncertainty-aware normal-pattern matching, and graph-based local structural modeling play distinct roles.

The main contributions of this paper are summarized as follows:

- We develop a multi-scale anomaly detection framework for time series with exogenous or auxiliary contextual inputs. Different from a simple multi-branch fusion design, the proposed framework assigns different modeling roles to the two temporal scales: the coarse-grained branch handles lag-enhanced global normal-pattern matching, whereas the fine-grained graph branch captures local structural deviations and channel-wise dependencies.
- We introduce an explicit lag-enhancement mechanism before uDTW-based matching. The lag alignment module estimates soft delayed responses from exogenous or contextual inputs to endogenous variables within a predefined lag range, and constructs lag-compensated contextual representations.

In this design, uDTW [20] is used only as an uncertainty-aware distance for matching lag-enhanced windows against the normal reference library, not as a direct lag estimator.

- We evaluate the method under three complementary input settings: proxy auxiliary context on MSDS, controlled native exogenous variables on the Linear Dataset, and operational context on SWaT. In addition to the main baseline comparison, we report MLP-enhanced baselines, DTW-variant replacement, auxiliary-input removal, single-scale and reversed scale-module variants, learned-lag analysis, reference-library size sensitivity, and computational cost.

The remainder of this paper is organized as follows. Section 2 reviews related work on time-series anomaly detection, exogenous variable modeling, application-domain monitoring, and DTW-based alignment methods. Section 3 presents the proposed EMS-uDTWAD framework and describes each module in detail. Section 4 reports the experimental setup, comparative results, ablation studies, robustness analysis, visualization results, and parameter sensitivity analysis. Section 5 concludes the paper and discusses future research directions.

2 Related Work

2.1 Time Series Anomaly Detection

Time-series anomaly detection has been studied in both univariate and multivariate forms. Existing methods include classical time-series similarity approaches [5], statistical and machine-learning outlier-detection methods [6–8], deep learning models [12,21,22], and time-series foundation models [23]. Prediction-based models estimate future values and detect large prediction errors; reconstruction-based models identify samples that cannot be well reconstructed; contrastive-learning methods enlarge the representation gap between normal and abnormal patterns. For example, CSLSTMs [21] uses noise decomposition and contextual information, CATCH [22] exploits frequency-wise patching and adaptive channel correlations, DCdetector [12] learns contrastive representations from latent temporal patterns, and MOMENT [23] provides a foundation-model-based time-series representation. Recent studies further improve robustness and multi-scale representation through bidirectional consistency discrimination for soft-sensor attack detection [15] and decomposition-based multi-scale Transformers [16]. However, these methods mainly model the observed endogenous variables. When a normal operating change is caused by a delayed contextual response, the detector may still assign a high anomaly score if the contextual input is not aligned with the endogenous response.

2.2 Exogenous Variable Modeling in Time Series

Exogenous variables are widely used in time-series forecasting to improve prediction under heterogeneous operating conditions. For example, building on the Transformer architecture [24], TimeXer [18] introduces exogenous variables into Transformer-based forecasting through tokenization and cross-attention, and GCGNet [25] jointly models temporal dependencies and variable correlations. However, these studies mainly focus on forecasting rather than anomaly detection. When exogenous effects appear with temporal delays, direct concatenation or synchronous modeling may fail to capture the delayed response between contextual inputs and endogenous variables. Therefore, anomaly detection models need to use auxiliary inputs in a lag-aware manner to distinguish true anomalies from normal delayed responses.

2.3 Application-Domain Monitoring with Contextual Inputs

Time-series anomaly detection is closely related to IoT and industrial monitoring. IoT systems continuously collect heterogeneous sensor streams, where abnormal patterns may arise from device faults,

communication instability, abnormal operation, or environmental changes [2,26,27]. In such monitoring settings, contextual factors such as operating state, workload, control actions, communication conditions, and environmental changes can influence the observed variables. However, most application-domain studies either use such context as synchronous covariates or focus on system-specific prediction and diagnosis. The delayed response between contextual inputs and monitored variables is still less explicitly modeled in anomaly detection, which motivates the lag-enhanced design of EMS-uDTWAD.

2.4 Dynamic Time Warping and Its Variants

Dynamic Time Warping (DTW) [19] is a sequence alignment method that establishes correspondences between elements of two sequences according to the minimum-distance principle, and it is commonly used to evaluate sequence similarity. However, DTW suffers from non-differentiability and sensitivity to noise, which limits its practical applicability. In contrast, soft dynamic time warping (sDTW) [28] introduces a smoothing parameter to define a differentiable soft-DTW distance, thereby alleviating, to some extent, the non-differentiability of DTW and its excessive sensitivity to noise. sDTW has achieved promising results in tasks such as action recognition and time series forecasting, yet it still has certain limitations. For example, noise and outliers may still cause sDTW to fall into poor local optima. Uncertainty-aware dynamic time warping, namely uDTW, incorporates uncertainty estimation into the dynamic time warping process to improve the robustness of sequence alignment [20]. Unlike the aforementioned methods, uDTW generates a corresponding variance for each time step to characterize uncertainty and constrains variance estimation through a regularization term, thereby reducing the interference of noise and outliers with sequence alignment results. Here, γ denotes the smoothing coefficient; a larger γ leads to weaker differences among path costs. Fig. 2a–d shows the alignment heatmaps of uDTW and sDTW under different smoothing coefficients γ. As γ increases, more candidate paths are incorporated into the computation, and the visualization gradually becomes blurred or even bifurcated. When γ is small, the selected path more closely resembles the hard alignment path of conventional DTW. In particular, the bifurcation phenomenon observed in the figure suggests that uDTW can preserve multiple plausible alignment paths in uncertain regions, which may help reduce the adverse effects of noise interference and local misalignment. In this study, uDTW is adopted for uncertainty-aware sequence matching between lag-enhanced windows, while exogenous-to-endogenous lag estimation is handled by the preceding lag alignment module. These observations motivate the controlled DTW-variant comparison reported later in Section 4.5.3.

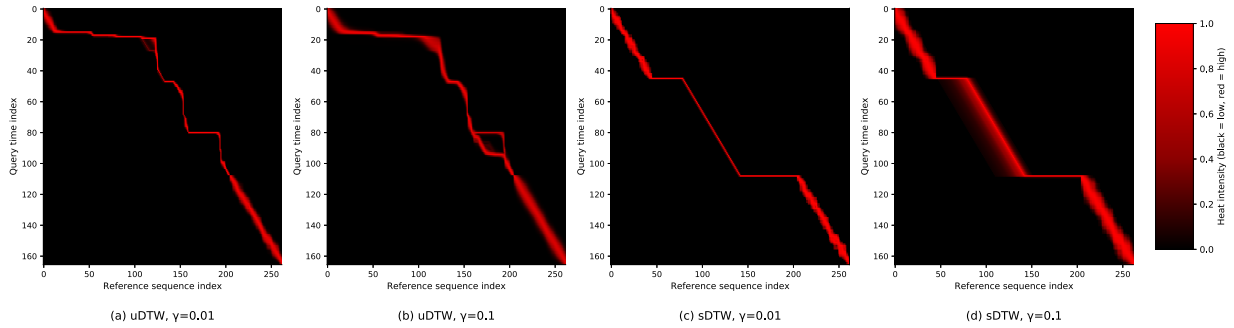

Figure 2: Comparison of alignment heatmaps generated by uDTW and sDTW with different values of the smoothing coefficient γ: (**a**) uDTW with $\gamma = 0.01$; (**b**) uDTW with $\gamma = 0.1$; (**c**) sDTW with $\gamma = 0.01$; (**d**) sDTW with $\gamma = 0.1$. Brighter red regions indicate higher normalized alignment intensity.

3 Methodology

In anomaly detection with exogenous or contextual inputs, given the endogenous observation window $\mathbf{X}_i^{\text{endo}} \in \mathbb{R}^{N\times T}$ and the exogenous input window $\mathbf{X}_i^{\text{exo}} \in \mathbb{R}^{D\times T}$, the objective is to construct an anomaly detection function that outputs an anomaly score $s_i \in \mathbb{R}$ and a window-level anomaly label $y_i \in \{0,1\}$. Here, N and D denote the numbers of endogenous and exogenous variables, respectively, and T denotes the window length. The detection function is formulated as

$$(s_i, y_i) = F\left(\mathbf{X}_i^{\text{endo}}, \mathbf{X}_i^{\text{exo}}\right). \tag{1}$$

For notational simplicity, $\mathbf{X}_i^{\text{exo}}$ denotes either native exogenous variables or auxiliary contextual inputs, depending on the dataset. When true exogenous variables are unavailable, as in MSDS, this term refers only to proxy auxiliary context rather than to an independent external driver. The interpretation of the learned delayed response is therefore dataset-dependent.

EMS-uDTWAD performs multi-scale anomaly modeling by separating three operations: exogenous-to-endogenous lag alignment, uDTW-based normal-pattern matching, and fine-grained graph-structural modeling. Fig. 3 illustrates the overall framework of the proposed method, which mainly consists of four modules: the multi-scale windowing module, the graph embedding module, the lag alignment and uDTW matching module, and the score fusion and anomaly decision module.

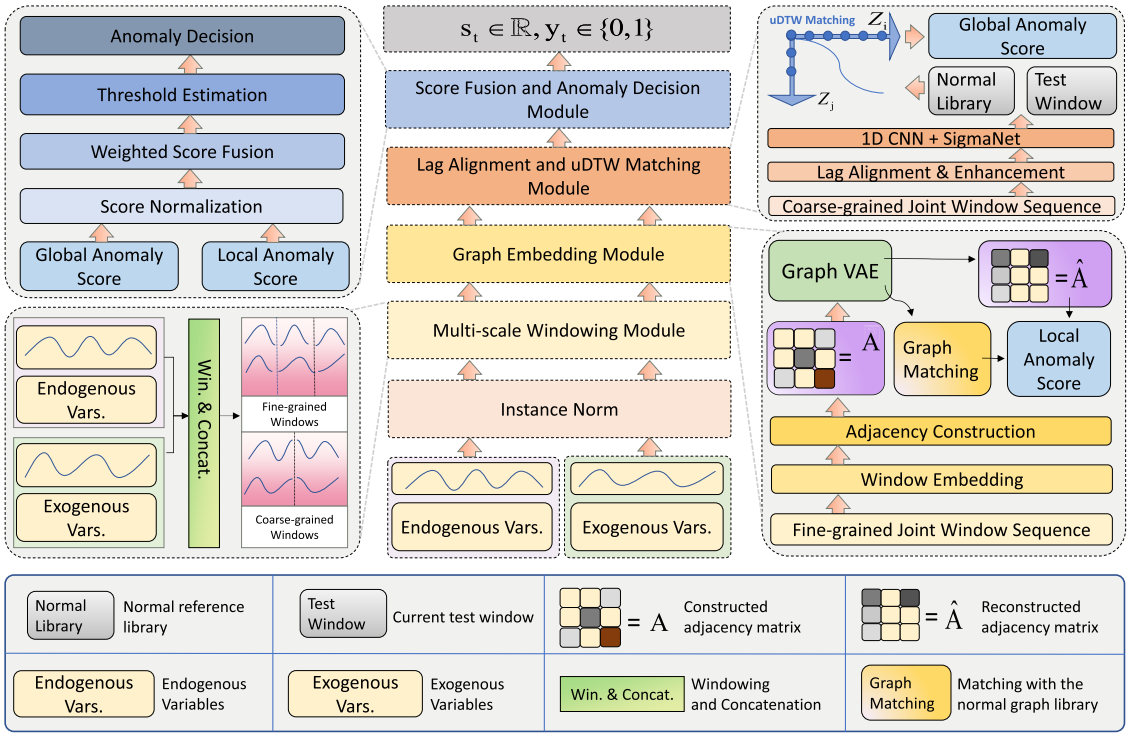

Figure 3: Overall framework of the proposed multi-scale time-series anomaly detection method. The coarse-grained branch first performs exogenous-to-endogenous lag alignment and enhancement, and then applies uDTW matching to compare the lag-enhanced test window with the normal reference library.

3.1 Overall Architecture Overview

EMS-uDTWAD first normalizes the endogenous and contextual inputs with instance normalization [29] and then generates right-aligned fine-grained and coarse-grained windows. The two scales are

used for different purposes. The fine-grained branch builds a learned adjacency representation from local joint windows and uses a Graph VAE to obtain the graph-structure anomaly score $s^{(\mathrm{G})}$. The coarse-grained branch first estimates delayed contextual responses within a predefined lag range. The resulting lag-enhanced windows are encoded by a 1D convolutional layer, processed by SigmaNet for time-step-level uncertainty estimation, and compared with the normal reference library using uDTW. The minimum normalized uDTW distance gives the global anomaly score $s_i^{(\mathrm{U})}$. The two normalized scores are then fused with a validation-selected weight, and the final threshold is estimated from normal validation samples.

3.2 Multi-Scale Windowing Module

The multi-scale windowing module aims to construct fine-grained and coarse-grained temporal representations of endogenous and exogenous variables, thereby providing multi-scale inputs for subsequent modeling. Specifically, instance normalization [29] is first adopted to reduce distributional differences between different variables. Then, the endogenous variable X_{endo} and the exogenous variable X_{exo} are fed into the sliding window generator.

Let the input time series be denoted as $X_{\mathrm{endo}} \in \mathbb{R}^{N\times L}$ and $X_{\mathrm{exo}} \in \mathbb{R}^{D\times L}$, respectively. The sliding window operation is performed using a window size W and a window stride S. Here, W is set to W_{fine} or W_{coarse} according to the temporal scale. Each window contains W consecutive time steps, and adjacent windows are separated by S time steps. After the sliding window operation, $\tilde{X}_{\mathrm{endo}} \in \mathbb{R}^{N\times W\times C}$ and $\tilde{X}_{\mathrm{exo}} \in \mathbb{R}^{D\times W\times C}$ can be obtained as follows:

$$\tilde{X}_{\mathrm{endo}} = \mathrm{Window}\left(X_{\mathrm{endo}}; W, S\right), \quad \tilde{X}_{\mathrm{exo}} = \mathrm{Window}\left(X_{\mathrm{exo}}; W, S\right), \tag{2}$$

where W denotes the window size, S denotes the window stride, and C denotes the number of generated windows. To characterize different temporal scales, W_{fine} is defined as the fine-grained window size, while W_{coarse} is defined as the coarse-grained window size. The relationship between them can be expressed as

$$W_{\mathrm{coarse}} = \lambda W_{\mathrm{fine}}, \quad \lambda > 1, \tag{3}$$

where λ is the scale factor, which is typically set to 2 or 4.

Subsequently, the windowed representations $\tilde{X}_{\mathrm{endo}}$ and $\tilde{X}_{\mathrm{exo}}$ are concatenated along the variable dimension. This process can be formulated as

$$\tilde{S} = \begin{pmatrix} \tilde{X}_{\mathrm{endo}} \\ \tilde{X}_{\mathrm{exo}} \end{pmatrix} \in \mathbb{R}^{(N+D)\times W\times C}, \tag{4}$$

where $\tilde{S}$ denotes the concatenated joint window sequence, with dimensions of $(N + D) \times W \times C$. Furthermore, $\tilde{S}$ can be regarded as consisting of C windows, where the c-th window is denoted as $\tilde{S}^{(c)} \in \mathbb{R}^{(N+D)\times W}$, $c = 1, 2, \ldots, C$. For different temporal scales, $W = W_{\mathrm{fine}}$ and $W = W_{\mathrm{coarse}}$ are respectively adopted, and the above sliding window and concatenation procedures are repeated. As a result, the fine-grained joint window sequence $\tilde{S}^{\mathrm{fine}}$ and the coarse-grained joint window sequence $\tilde{S}^{\mathrm{coarse}}$ are obtained. These two sequences are then fed into the graph embedding module and the lag alignment and uDTW matching module for fine-grained and coarse-grained anomaly modeling, respectively.

To ensure score-level correspondence between the two branches, this study adopts a right-aligned multi-scale window pairing strategy. The fine-grained and coarse-grained windows are aligned to the same ending time point instead of sharing the same starting index. In this way, the fine-grained branch describes recent local variations, while the coarse-grained branch provides longer contextual information for the same detection position.

Specifically, the ending index of the c-th paired sample is defined as

$$b_c = W_{\text{coarse}} + (c-1)S, \quad c = 1, 2, \ldots, C. \tag{5}$$

The corresponding coarse-grained window and fine-grained window are defined as

$$\mathcal{W}_c^{\text{coarse}} = \{b_c - W_{\text{coarse}} + 1, \ldots, b_c\}, \tag{6}$$

and

$$\mathcal{W}_c^{\text{fine}} = \{b_c - W_{\text{fine}} + 1, \ldots, b_c\}. \tag{7}$$

The number of paired windows is determined by the coarse-grained window size:

$$C = \left\lfloor \frac{L - W_{\text{coarse}}}{S} \right\rfloor + 1. \tag{8}$$

Thus, the c-th fine-grained window and the c-th coarse-grained window correspond to the same ending time point b_c. This pairing strategy provides a clearer correspondence between the two branch-level scores and does not introduce future information.

3.3 Graph Embedding Module

The graph embedding module is responsible for channel-wise dependency modeling in EMS-uDTWAD. It captures fine-grained structural dependencies among endogenous and exogenous variables through a learned adjacency representation, thereby enabling fine-grained anomaly detection. In this module, each node corresponds to a variable representation within a fine-grained window, and the edge weights describe the latent dependency strength between different variables in the same window. First, each fine-grained joint window $(\tilde{S}^{\text{fine}})^{(c)}$ is fed into an embedding layer, where each variable within the current window is mapped into a vector representation. Let $H_{\text{fine}}^{(c)} \in \mathbb{R}^{M \times d_g}$ denote the node embedding matrix of the c-th fine-grained window, where $M = N + D$ is the number of endogenous and exogenous variables, and d_g is the embedding dimension. The embedding representation is obtained as follows:

$$H_{\text{fine}}^{(c)} = \text{Embedding}\left((\tilde{S}^{\text{fine}})^{(c)}\right). \tag{9}$$

Subsequently, an adjacency matrix is constructed based on the above vector representation to jointly characterize temporal dependencies and channel-wise correlations:

$$A_{\text{raw}}^{(c)} = \text{GELU}\left(\left(H_{\text{fine}}^{(c)} W_1\right)\left(H_{\text{fine}}^{(c)} W_2\right)^{\text{T}}\right), \tag{10}$$

where $W_1, W_2 \in \mathbb{R}^{d_g \times d_a}$ are learnable projection matrices, and $A_{\text{raw}}^{(c)} \in \mathbb{R}^{M \times M}$ represents the learned dependency matrix among variables in the c-th fine-grained window.

To ensure the symmetry of the graph structure, the adjacency matrix is symmetrized as follows:

$$\tilde{A}^{(c)} = \frac{1}{2}\left(A_{\text{raw}}^{(c)} + \left(A_{\text{raw}}^{(c)}\right)^{\text{T}}\right). \tag{11}$$

Thus, the matrix $\tilde{A}^{(c)}$ is obtained. The elements of $\tilde{A}^{(c)}$ describe the latent dependencies among variables within the c-th fine-grained window, thereby reflecting the local temporal patterns encoded by each variable

representation and the cross-channel correlations among variables. Furthermore, $\tilde{A}^{(c)}$ is fed into a graph variational autoencoder (Graph VAE) [30] to obtain the reconstructed graph $\hat{A}^{(c)}$. The reconstruction loss is defined as

$$\mathcal{L}_m^{(c)} = \left\| \tilde{A}^{(c)} - \hat{A}^{(c)} \right\|_1 . \tag{12}$$

The Kullback–Leibler (KL) divergence regularization term is defined as

$$\mathcal{L}_{\mathrm{KL}}^{\mathrm{G},(c)} = D_{\mathrm{KL}} \left(q\left(Z^{A,(c)} \mid \tilde{A}^{(c)} \right) \parallel p\left(Z^{A,(c)} \right) \right). \tag{13}$$

The total loss function is defined as

$$\mathcal{L}_{\mathrm{G}} = \frac{1}{C} \sum_{c=1}^{C} \left(\mathcal{L}_m^{(c)} + \eta \mathcal{L}_{\mathrm{KL}}^{\mathrm{G},(c)} \right), \tag{14}$$

where η is the weight coefficient used to balance the reconstruction loss and the KL-divergence regularization term. By minimizing the loss function $\mathcal{L}_{\mathrm{G}}$, the graph embedding module and the Graph VAE module can be jointly trained.

In the testing stage, a library of reconstructed normal graphs is first constructed using normal training windows. Then, the reconstructed graphs generated from the test set are matched with this library through nearest-neighbor matching. The anomaly score of the graph branch is defined as

$$s_i^{(\mathrm{G})} = \min_r \left\| \hat{A}_i^{(\mathrm{test})} - \hat{A}_r^{(\mathrm{train})} \right\|_{\mathrm{F}} . \tag{15}$$

Finally, the distribution of anomaly scores is estimated on normal samples from the validation set, and the anomaly decision threshold is determined accordingly.

3.4 Lag Alignment and uDTW Matching Module

This module contains two consecutive operations: lag alignment and uDTW-based matching. The lag alignment step first estimates delayed responses from exogenous channels to endogenous channels at the coarse-grained scale and constructs lag-enhanced joint windows. The uDTW step then compares each lag-enhanced test window with lag-enhanced normal windows in the reference library. Thus, uDTW is used to compute an uncertainty-aware deviation score, rather than to estimate inter-channel lags or construct channel-wise dependencies.

For the c-th coarse-grained window generated in Section 3.2, the endogenous and exogenous components are denoted as

$$\mathbf{X}_{\mathrm{endo}}^{(c)} \in \mathbb{R}^{N \times W_{\mathrm{coarse}}}, \qquad \mathbf{X}_{\mathrm{exo}}^{(c)} \in \mathbb{R}^{D \times W_{\mathrm{coarse}}}, \tag{16}$$

where N and D denote the numbers of endogenous and exogenous variables, respectively. To represent possible delayed responses from exogenous variables to endogenous variables, a maximum lag $K_{\max}$ is introduced, and the candidate lag set is defined as

$$\mathcal{K} = \{0, 1, \ldots, K_{\max}\}. \tag{17}$$

For each exogenous channel d and candidate lag k, the shifted exogenous sequence is defined as

$$\mathbf{x}_{d,k}^{\mathrm{exo},(c)} = \mathrm{Shift}\left(\mathbf{x}_{d}^{\mathrm{exo},(c)}, k\right), \tag{18}$$

where $\mathrm{Shift}(\cdot, k)$ shifts the exogenous sequence backward by k time steps. Boundary positions introduced by shifting are handled using a temporal mask to avoid using invalid observations. The shifting operation uses only the exogenous observations within the current coarse-grained window and does not introduce future information beyond the detection endpoint.

For each endogenous channel n, a lag-relevance score is computed between the endogenous sequence and each lagged exogenous candidate:

$$e_{n,d,k}^{(c)} = g_{\theta}\left(\mathbf{x}_{n}^{\mathrm{endo},(c)}, \mathbf{x}_{d,k}^{\mathrm{exo},(c)}\right), \tag{19}$$

where $g_{\theta}(\cdot)$ denotes a learnable lag-relevance scoring function. To avoid losing the temporal correspondence introduced by the candidate lag k, the two sequences are not pooled independently. Instead, for each valid time position, the endogenous value and the lag-shifted exogenous value are first paired as $(x_{n,t}^{\mathrm{endo},(c)}, x_{d,t-k}^{\mathrm{exo},(c)})$. Lag-specific interaction features, including the paired values, their absolute difference, and their product, are then aggregated over valid positions and fed into a lightweight MLP to produce $e_{n,d,k}^{(c)}$. In this way, the score is computed from lag-dependent paired observations rather than from shift-invariant marginal summaries. The corresponding lag weights are obtained by a softmax operation:

$$a_{n,d,k}^{(c)} = \frac{\exp\left(e_{n,d,k}^{(c)}\right)}{\sum_{d'=1}^{D}\sum_{k'=0}^{K_{\max}}\exp\left(e_{n,d',k'}^{(c)}\right)}. \tag{20}$$

To make the estimated lag relationship explicit for each exogenous–endogenous channel pair, the lag weights are further normalized over the candidate lag set:

$$\bar{a}_{n,d,k}^{(c)} = \frac{a_{n,d,k}^{(c)}}{\sum_{k'=0}^{K_{\max}} a_{n,d,k'}^{(c)} + \varepsilon}. \tag{21}$$

The soft lag from the d-th exogenous channel to the n-th endogenous channel in the c-th window is then defined as

$$\hat{\tau}_{n,d}^{(c)} = \sum_{k=0}^{K_{\max}} k \cdot \bar{a}_{n,d,k}^{(c)}. \tag{22}$$

The value of $\hat{\tau}_{n,d}^{(c)}$ gives an interpretable estimate of the delayed response associated with this exogenous–endogenous channel pair. These lag weights are then used to form a lag-compensated exogenous response. For each endogenous channel, the shifted exogenous candidates are aggregated using their learned weights, so that the subsequent uDTW branch receives a delayed-response representation rather than a purely synchronous concatenation.

Based on the learned lag weights, the lag-compensated exogenous response associated with the n-th endogenous channel is constructed as

$$r_{n,t}^{(c)} = \sum_{d=1}^{D}\sum_{k=0}^{K_{\max}} a_{n,d,k}^{(c)} m_{t,k} x_{d,t-k}^{\mathrm{exo},(c)}, \tag{23}$$

where $m_{t,k} \in \{0,1\}$ is the temporal mask indicating whether the shifted value $x_{d,t-k}^{\mathrm{exo},(c)}$ is valid. By stacking the lag-compensated responses of all endogenous channels, the lag-compensated exogenous representation is obtained as

$$\mathbf{R}_{\mathrm{lag}}^{(c)} = \left[r_{n,t}^{(c)} \right]_{n=1,t=1}^{N, W_{\mathrm{coarse}}} \in \mathbb{R}^{N \times W_{\mathrm{coarse}}}. \tag{24}$$

The final lag-enhanced coarse-grained joint window is constructed by concatenating the endogenous window, the original exogenous window, and the lag-compensated exogenous response:

$$\mathbf{S}_{\mathrm{lag}}^{(c)} = \mathrm{Concat}\left(\mathbf{X}_{\mathrm{endo}}^{(c)}, \mathbf{X}_{\mathrm{exo}}^{(c)}, \mathbf{R}_{\mathrm{lag}}^{(c)} \right). \tag{25}$$

This representation preserves the original observations while explicitly incorporating the delayed exogenous response associated with endogenous variables.

Specifically, for the lag-enhanced coarse-grained joint window $\mathbf{S}_{\mathrm{lag}}^{(c)}$, this study adopts a window-wise convolution strategy to extract local temporal contextual features, thereby providing more discriminative semantic representations for subsequent uDTW-based matching. A one-dimensional convolution operation with shared parameters is applied as follows:

$$H^{(c)} = \mathrm{Conv1D}\left(\mathbf{S}_{\mathrm{lag}}^{(c)} \right), \quad H^{(c)} \in \mathbb{R}^{p' \times d}, \quad c = 1, 2, \ldots, C, \tag{26}$$

where d denotes the number of output channels of the convolutional layer, and p' denotes the temporal length after convolution, which is determined by the kernel size, stride, and padding strategy. Since all windows share the same set of convolutional parameters, the above operation can learn consistent local temporal representations across different time windows. To preserve time-step-level information and maintain consistency with the subsequent uncertainty estimation and uDTW-based matching process, this study does not perform pooling compression along the temporal dimension. Instead, the convolutional output of the c-th window is directly used as its latent sequential feature representation, denoted as

$$Z^{(c)} = H^{(c)}, \quad Z^{(c)} \in \mathbb{R}^{p' \times d}, \quad c = 1, 2, \ldots, C, \tag{27}$$

where $Z^{(c)}$ represents the latent time-series feature representation of the c-th window after one-dimensional convolution, and each row corresponds to the feature vector of one time step.

Subsequently, the SigmaNet uncertainty estimation module incorporates time-step-level uncertainty into the feature modeling process. For each time step in the c-th window, it estimates a d-dimensional uncertainty representation, resulting in the time-step-level uncertainty representation $U^{(c)}$:

$$U^{(c)} = \mathrm{SigmaNet}\left(Z^{(c)} \right), \quad U^{(c)} \in \mathbb{R}^{p' \times d}, \quad c = 1, 2, \ldots, C, \tag{28}$$

where $U^{(c)}$ is temporally aligned with the latent feature $Z^{(c)}$, and is used to characterize the uncertainty level of the feature representation at each time step. For any pair of input sequences, this study further constructs a paired uncertainty matrix Σ based on their time-step-level uncertainty representations. This matrix is combined with the distance matrix between latent features to perform uncertainty-aware uDTW-based matching.

After obtaining the latent feature $Z^{(c)}$ and the time-step-level uncertainty representation $U^{(c)}$ of the c-th window, this study further constructs the paired uncertainty matrix Σ. The transformed uncertainty matrix and the feature distance matrix are then jointly fed into the uDTW module. During training, the distance loss

$\mathcal{L}_d$ and the uncertainty regularization term $\mathcal{L}_s$ produced by uDTW are used to jointly optimize the feature extraction network and the uncertainty estimation network. In this way, the model can learn a more stable representation for uncertainty-aware matching between lag-enhanced coarse-grained windows. Following the uncertainty-aware DTW formulation in uDTW [20], the similarity learning objective is formulated as follows:

$$\min_{\rho,\rho_\sigma} \sum_q \ell\left(\mathrm{uDTW}\left(D\left(\Psi_q, \Psi_q'\right), \Sigma_{\mathrm{w}}\left(\Psi_q, \Psi_q'\right)\right), \delta_q\right) + \beta\Omega\left(\Sigma\left(\Psi_q, \Psi_q'\right)\right), \tag{29}$$

where $\Psi_q = \mathrm{Conv1D}(\mathbf{S}_{\mathrm{lag}}^{(q)}; \rho)$ and $\Psi_q' = \mathrm{Conv1D}(\mathbf{S}_{\mathrm{lag}}'^{(q)}; \rho)$ denote the feature representations of the q-th pair of lag-enhanced windows obtained by the feature extraction network, respectively. δ_q denotes the similarity indicator of the sample pair, and β is the weight coefficient of the uncertainty regularization term. $D(\Psi_q, \Psi_q')$ denotes the distance matrix constructed from the two sequence feature representations. $\Sigma(\Psi_q, \Psi_q')$ denotes the corresponding uncertainty matrix, and $\Sigma_{\mathrm{w}}(\Psi_q, \Psi_q')$ denotes the transformed uncertainty matrix used to weight the local matching cost in the uDTW-based matching process. In this study, the sample-pair indicator δ_q is constructed only from normal training windows. Two windows are treated as a similar pair if their starting indices are within a predefined temporal neighborhood or if one window is generated from the other through data augmentation. Windows sampled from distant temporal positions are regarded as dissimilar pairs. This pair construction process does not use anomaly labels, thereby avoiding label leakage.

This objective jointly optimizes the uDTW matching distance and uncertainty regularization.

The transformation from Σ to Σ_{w} is used to ensure that the uncertainty weights remain positive and numerically stable during uDTW-based matching.

For the construction of Σ, this study follows the independent variance modeling strategy in uDTW [20] for different time points, which is defined as

$$\Sigma = \frac{1}{2}\left[\sigma^2\left(v_m; \rho_\sigma\right) + \sigma^2\left(v_n'; \rho_\sigma\right)\right]_{(m,n)\in I_\tau \times I_{\tau'}}, \tag{30}$$

where I_τ and $I_{\tau'}$ denote the temporal index sets corresponding to the two sequences, respectively. v_m and v_n' denote the feature vectors of the two sequences at the m-th and n-th time steps, respectively. The function $\sigma^2(\cdot; \rho_\sigma)$ denotes the variance estimation mapping implemented by SigmaNet, which takes the time-step feature vector as input and outputs the corresponding uncertainty estimation. This study assumes that the variances at different time points are mutually independent, and constructs the overall uncertainty matrix Σ accordingly.

In the testing stage, a normal reference library is constructed using lag-enhanced windows extracted from the normal training set. Each lag-enhanced test window is then matched with every reference window in the normal reference library, and the minimum normalized uDTW distance is taken as the global anomaly score. The final anomaly score is defined as

$$s_i^{(\mathrm{U})} = \min_{\tau_j^{\mathrm{lag}} \in \mathcal{R}_{\mathrm{lag}}} \frac{\mathrm{uDTW}\left(\tau_i^{\mathrm{lag}}, \tau_j^{\mathrm{lag}}\right)}{L_{i,j}}, \tag{31}$$

where $\mathcal{R}_{\mathrm{lag}}$ denotes the normal reference library constructed from lag-enhanced normal windows, τ_i^{lag} denotes the i-th lag-enhanced test window, and τ_j^{lag} denotes the j-th lag-enhanced normal window in the reference library. $L_{i,j}$ denotes the length of the uDTW matching path between τ_i^{lag} and τ_j^{lag}, which is used to normalize the uDTW distance. For simplicity, $\mathrm{uDTW}(\tau_i^{\mathrm{lag}}, \tau_j^{\mathrm{lag}})$ denotes the weighted matching

distance calculated after lag enhancement, feature extraction, and uncertainty estimation. After obtaining the anomaly score, the final anomaly decision is made using a threshold estimator. Accordingly, $s_i^{(\mathrm{U})}$ measures the deviation of a lag-enhanced test window from the normal reference library. The uDTW operation is not used to estimate the exogenous-to-endogenous lag; that role is assigned to the preceding lag alignment step.

3.5 Score Fusion and Anomaly Decision Module

Within the proposed framework, the uDTW branch is designed to describe global normal-pattern deviations in lag-enhanced coarse-grained windows, whereas the graph embedding branch is introduced to capture fine-grained structural deviations in temporal interactions and cross-channel correlations. Since these two branches characterize anomalous behavior from global and local perspectives, respectively, they provide complementary information. Accordingly, their anomaly scores are integrated to enhance the stability and reliability of the final decision.

Let $s_i^{(\mathrm{U})}$ denote the global anomaly score produced by the uDTW branch for the i-th test window, and let $s_i^{(\mathrm{G})}$ denote the local anomaly score generated by the graph embedding branch. Because the two scores may lie on different numerical scales, they are first normalized as follows:

$$\tilde{s}_i^{(\mathrm{U})} = \mathrm{Norm}\left(s_i^{(\mathrm{U})}\right), \qquad \tilde{s}_i^{(\mathrm{G})} = \mathrm{Norm}\left(s_i^{(\mathrm{G})}\right), \tag{32}$$

where $\mathrm{Norm}(\cdot)$ denotes min–max normalization computed from the statistics of normal samples in the validation set, which is defined as

$$\mathrm{Norm}(x) = \frac{x - x_{\min}}{x_{\max} - x_{\min} + \varepsilon}, \tag{33}$$

where $x_{\min}$ and $x_{\max}$ represent the minimum and maximum values of the corresponding branch on normal samples in the validation set, respectively, and ε is a small constant introduced to avoid division by zero.

After normalization, the global and local scores are combined through weighted summation, yielding the final anomaly score for the i-th test window:

$$s_i = \alpha \tilde{s}_i^{(\mathrm{U})} + (1-\alpha)\tilde{s}_i^{(\mathrm{G})}, \qquad \alpha \in [0,1], \tag{34}$$

where α controls the relative contributions of the global and local branches. In the main experiments, α is selected from $\{0, 0.1, \ldots, 1.0\}$ on the validation set for each dataset and is then fixed before test evaluation.

For anomaly decision, the distribution of fused scores on normal validation samples is first estimated, and the decision threshold is then determined using a high-percentile rule. Let $\{s_k\}_{k\in\mathrm{V}}$ denote the set of fused anomaly scores associated with normal samples in the validation set, where $\mathcal{V}$ denotes the index set of normal validation samples. The threshold θ is defined as

$$\theta = \mathrm{Percentile}_q(\{s_k\}_{k\in\mathcal{V}}), \tag{35}$$

where $\mathrm{Percentile}_q(\cdot)$ denotes the q-th percentile of the score distribution. In practical settings, q is typically chosen as 95 or 99.

Once θ is obtained, the anomaly label is assigned according to the comparison between the final score and the threshold:

$$y_i = \begin{cases} 1, & s_i > \theta, \\ 0, & s_i \le \theta, \end{cases} \tag{36}$$

where $y_i = 1$ indicates that the i-th test window is classified as anomalous, while $y_i = 0$ indicates a normal window.

Because anomaly scores are initially produced at the window level, they are further mapped back to the original time axis according to the temporal span covered by each paired multi-scale window. For the i-th paired window, let b_i be its ending index. Its temporal range is then defined based on the coarse-grained window as

$$\mathcal{T}_i = \{b_i - W_{\text{coarse}} + 1, \ldots, b_i\}. \tag{37}$$

For any time point t, let $\mathcal{I}(t)$ denote the set of windows that cover t:

$$\mathcal{I}(t) = \{\, i \mid t \in \mathcal{T}_i \,\}. \tag{38}$$

The point-wise anomaly score is obtained by aggregating the scores of all windows covering that time point in a weighted manner:

$$s_t = \frac{\sum_{i \in \mathcal{I}(t)} \omega_{i,t} s_i}{\sum_{i \in \mathcal{I}(t)} \omega_{i,t} + \varepsilon}, \tag{39}$$

where $\omega_{i,t}$ denotes the contribution of the i-th window to time point t, and ε is a small constant used to avoid division by zero. Boundary time points not covered by any coarse-grained window are excluded from point-level evaluation. In this work, the weight is determined according to the distance between t and the center of the corresponding coarse-grained window:

$$\omega_{i,t} = 1 - \frac{|t - m_i|}{r_i + 1}, \qquad m_i = b_i - \frac{W_{\text{coarse}} - 1}{2}, \qquad r_i = \frac{W_{\text{coarse}} - 1}{2}. \tag{40}$$

Finally, the point-level threshold θ_p is estimated from the aggregated point-wise anomaly scores of normal validation samples, and the final point-level anomaly label is determined as

$$y_t = \begin{cases} 1, & s_t > \theta_p, \\ 0, & s_t \le \theta_p. \end{cases} \tag{41}$$

4 Experiments and Results

This section evaluates EMS-uDTWAD on two real-world datasets and one synthetic dataset. We describe the datasets, baselines, evaluation protocol, implementation details, comparative results, and model analyses.

4.1 Datasets

The three datasets cover complementary settings, as summarized in Table 1. MSDS provides a real-system proxy auxiliary-context setting, the Linear Dataset provides controlled native exogenous variables with anomalous interventions, and SWaT provides real industrial operational-context inputs.

This dataset design is intended to cover three complementary evaluation conditions rather than to claim that all real-world datasets contain native exogenous-variable annotations. The Linear Dataset provides a controlled setting with known exogenous drivers, MSDS evaluates whether proxy auxiliary context can

improve detection on a real system dataset, and SWaT evaluates whether operational control-state information can be exploited in a real industrial process. This distinction is important because the contribution of each auxiliary input should be interpreted according to its source and role in the corresponding dataset.

Table 1: Characterization of the datasets used in this study.

Dataset	Type	Endogenous	Context Input	Role
MSDS	Real system data	System metrics	Proxy auxiliary context	Proxy-context setting
Linear Dataset	Synthetic data	Simulated endogenous series	Native exogenous variables	Controlled native-exogenous setting
SWaT	Real industrial process data	Process measurements	Actuator/control-state variables	Operational-context setting

Note: MSDS does not provide native exogenous-variable columns and is used only as a proxy auxili ary-context setting. SWaT does not provide official exogenous-variable annotations, but its actuator and control-state variables provide real operational context for industrial anomaly detection.

The MSDS (Multi-Source Distributed System) dataset was constructed by Nedelkoski et al. [31] based on an OpenStack testbed, in which fault-injected samples are labeled as anomalies. Since MSDS does not provide native exogenous-variable columns, this study uses it only as a real-world proxy auxiliary-context setting. The constructed load-fluctuation proxy is intended to represent auxiliary contextual changes rather than a genuine external driving variable.

The Linear Dataset [32] exhibits linear interactive dynamics. Its endogenous variables are driven by corresponding exogenous variables, and anomalous samples are generated by injecting point anomalies and sequential anomalies. This dataset is used to evaluate the proposed method under a controlled setting with native exogenous variables.

The SWaT dataset [33] is collected from a six-stage water treatment testbed and contains both normal operation data and attack data. In this study, continuous process measurements, such as flow, level, pressure, and analyzer readings, are treated as endogenous variables, whereas actuator and control-state variables, such as pumps, valves, and ultraviolet devices, are used as operational contextual inputs. These variables are not treated as officially annotated exogenous variables; instead, they provide operational context for evaluating whether the proposed model can exploit control-state information in a real industrial process.

4.2 Baseline Methods and Evaluation Metrics

The proposed model is compared with six representative methods, including Informer [13], TFAD [34], Anomaly-Transformer [14], FCVAE [9], KAN-AD [35], and sDTW [28]. Among them, Informer is an attention-based forecasting model; TFAD is an analytical model that integrates time-domain and frequency-domain modeling; Anomaly-Transformer applies the Transformer architecture to unsupervised outlier detection; FCVAE reconstructs sequences through a frequency-conditioned variational autoencoder; KAN-AD is a lightweight forecasting-based model; and sDTW alleviates the difficulty of backpropagation in DTW by introducing a smoothing parameter.

For baseline comparison, two versions of each baseline are evaluated. The first version follows the original input setting and model structure of the corresponding method. The second version, denoted as "+MLP", uses a lightweight two-layer MLP to fuse the auxiliary contextual input with the endogenous

representation before it is fed into the detection model. This setting allows the baselines to access the same type of auxiliary input as EMS-uDTWAD, but it is reported separately because it changes the input representation of the original baseline.

The original and MLP-enhanced versions are tuned separately using the same training, validation, and test split. Hyperparameters, early stopping, and threshold selection are determined only on the validation set, and the selected configuration is fixed before test evaluation. This validation protocol is applied to all baselines and to both original and MLP-enhanced versions. No test labels are used for model selection.

Precision, Recall, and F1-score are used for evaluation, and F1-score is treated as the primary metric. Therefore, the main comparison and model analysis tables report F1 scores for compactness.

To ensure consistency with commonly used evaluation protocols in time-series anomaly detection, the point-adjust technique [36] is adopted. According to this rule, if any time point within an anomalous segment is correctly identified, all anomalous points in that segment are regarded as successfully detected. It should be noted that point-adjusted evaluation may lead to relatively optimistic performance estimates; therefore, all compared methods are evaluated under the same protocol to ensure fairness.

4.3 Implementation Details

The proposed method was implemented in PyTorch [37] and trained using the Adam optimizer [38] with a learning rate of 1×10^{-3}. All experiments were conducted on a workstation equipped with an NVIDIA A800 GPU. Inference time was measured after model loading under the same software environment, excluding data preprocessing, file I/O, and visualization. For EMS-uDTWAD, reference-library features were precomputed from normal training windows before inference. Memory cost was measured by considering model parameters, cached reference-library representations, and peak GPU memory during test-window inference. For the MSDS, Linear Dataset, and SWaT, the fine-grained window size was set to $W_{\text{fine}} = 32$, and the coarse-grained window size was set to $W_{\text{coarse}} = 128$. The window stride was set to $S = 16$ for both fine-grained and coarse-grained window generation. Fine-grained and coarse-grained windows were paired using the right-aligned strategy described in Section 3.2, so that both branches produced anomaly scores for the same ending time point. For the baseline experiments, both the original baseline version and the corresponding MLP-enhanced version are implemented. In the original version, each baseline follows its original input setting and model structure. In the MLP-enhanced version, the endogenous representation and the auxiliary contextual representation are concatenated along the feature dimension and then passed through a two-layer MLP fusion module. The MLP contains one hidden layer, a ReLU activation function, and an output projection layer. The hidden dimension is selected from $\{32, 64, 128\}$ on the validation set.

For each baseline, the original version and the MLP-enhanced version are tuned independently under the same validation protocol. The learning rate is selected from $\{1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}\}$, the batch size is selected from $\{32, 64, 128\}$, and the threshold percentile q is selected from $\{95, 99\}$. The maximum number of training epochs is set to 100, with early stopping based on the validation loss and a patience of 10 epochs. After validation-based selection, the chosen configuration is fixed and evaluated on the test set. No test labels are used for model selection or threshold determination. Since $S < W_{\text{fine}} < W_{\text{coarse}}$, adjacent windows overlap, which helps preserve temporal continuity and allows window-level anomaly scores to be projected back to the original time axis. During inference, point-level anomaly scores are obtained by weighted aggregation of overlapping window scores, and the point-level decision threshold is estimated from the validation set. This setting corresponds to $\lambda = 4$, and the effect of alternative window-size combinations is further examined in the parameter sensitivity analysis. The weight coefficient of the KL regularization term, η, was set to 0.1, while the trade-off coefficient of the uncertainty regularization term, β, was set to 0.1 to control the contribution of the uncertainty regularization term to the overall loss. The branch fusion

coefficient α was selected from $\{0, 0.1, \ldots, 1.0\}$ based on the validation-set F1 score for each dataset. The selected values were $\alpha = 0.7$ for MSDS and $\alpha = 0.6$ for the Linear Dataset and SWaT. The effect of α is further reported in the sensitivity analysis. All hyperparameters were selected based only on the validation set, and the selected configuration was fixed before evaluation on the test set.

From the perspective of the structural causal model (SCM), this study defines endogenous and exogenous variables as follows. The observable multivariate time series is regarded as the endogenous variable, whereas external disturbance factors that drive system evolution are regarded as exogenous variables. For the MSDS dataset, the original data do not explicitly provide independent exogenous-variable columns. Therefore, this study constructs a proxy auxiliary-context variable to reflect short-term system-level fluctuations. This proxy is not regarded as a genuine external driving variable, but is used only to evaluate whether auxiliary contextual information can improve anomaly detection on a real-world system dataset. Specifically, the observable system metrics are first normalized using the statistics of the training set. Let $\tilde{x}_t^{(j)}$ denote the normalized value of the j-th system metric at time step t. The load-fluctuation proxy v_t is calculated as

$$v_t = \frac{1}{N} \sum_{j=1}^{N} \left| \tilde{x}_t^{(j)} - \tilde{x}_{t-1}^{(j)} \right|, \quad t = 2, \ldots, L, \tag{42}$$

where N denotes the number of observable system metrics. For the first time step, v_1 is set to 0. This proxy is used only to represent short-term system-level fluctuation derived from observable metrics; it is not an independent external driver. It does not use anomaly labels or fault-injection information. Therefore, label leakage is avoided during model training and evaluation. The 10-dimensional observable system metric sequences in MSDS are treated as endogenous variables, while the constructed load-fluctuation proxy is treated as proxy auxiliary context. For the Linear Dataset, the endogenous variables are defined as $x_t^{(1)}$, $x_t^{(2)}$, $x_t^{(3)}$, and $x_t^{(4)}$, and the exogenous variables are defined as $v_t^{(1)}$, $v_t^{(2)}$, $v_t^{(3)}$, and $v_t^{(4)}$. The anomaly term $\varepsilon_t^{(i)}$ represents an anomalous intervention on the exogenous variables.

For SWaT, the normal-operation data are used for training and validation, and the attack data are used for testing. The training portion is further divided into training and validation subsets at a ratio of 8:2. The "Normal/Attack" labels are used only for evaluation and are not used during model training. Continuous sensor measurements are treated as endogenous variables, while actuator and control-state variables are treated as operational contextual inputs. All continuous variables are normalized using the statistics of the training set, and the same normalization parameters are applied to the validation and test sets. Discrete actuator-state variables are retained as categorical or binary operational context features.

For the DTW-variant comparison, three versions of the proposed framework were evaluated: EMS-DTWAD, EMS-sDTWAD, and EMS-uDTWAD. They share the same window sizes, lag alignment module, graph embedding module, score fusion strategy, and threshold estimation. EMS-DTWAD uses conventional DTW in the coarse-grained matching branch, EMS-sDTWAD uses sDTW, and EMS-uDTWAD uses uDTW with time-step-level uncertainty estimation.

For the auxiliary-input removal analysis, an endogenous-only variant was evaluated. In this variant, the exogenous or auxiliary contextual inputs are removed from both branches. The graph branch constructs the adjacency representation using only endogenous variables, and the coarse-grained branch performs reference-library matching using only endogenous windows. The lag alignment step is disabled because no exogenous candidate sequence is available. The data split, window sizes, score fusion strategy, and threshold estimation are kept the same as those of EMS-uDTWAD.

For the MLP contextual-fusion removal analysis, the lightweight MLP-based contextual transformation is removed and replaced by direct concatenation followed by a linear projection. The remaining lag alignment, uDTW matching, graph embedding, score fusion, and threshold estimation procedures are kept unchanged. This variant is used to test whether nonlinear contextual fusion contributes to the final detection performance.

For the single-scale framework comparison, both branches are retained but the fine/coarse temporal separation is removed. Specifically, the graph embedding branch and the lag alignment with uDTW matching branch use the same window length $W_{\text{single}} = 64$, while the stride, auxiliary-input setting, score fusion strategy, validation protocol, and threshold estimation are kept identical to those of EMS-uDTWAD. This variant is used to examine whether the performance gain comes from the multi-scale temporal design rather than only from using two detection branches.

For the module-scale pairing analysis, a reversed scale-module variant was evaluated. In this variant, the lag alignment and uDTW matching branch is applied to fine-grained windows, while the graph embedding module is applied to coarse-grained windows. The data split, window sizes, score fusion strategy, threshold estimation, and validation protocol are kept the same as those of EMS-uDTWAD.

For the normal-reference-library size analysis, the size of $\mathcal{R}_{\text{lag}}$ was varied by retaining 25%, 50%, 75%, and 100% of the normal reference windows. The reference windows were selected only from the normal training set, and the validation set was used to determine the final configuration. No test labels were used in reference-library construction or model selection.

For the learned-lag analysis, the soft lag estimate $\hat{\tau}_{n,d}^{(c)}$ defined in Section 3.4 was recorded for each paired endogenous–contextual channel and each coarse-grained window. For the Linear Dataset, a controlled diagnostic setting was additionally constructed by assigning known delays between selected exogenous and endogenous channels. The learned soft lags were then compared with the controlled lag values using the mean absolute lag error. This diagnostic experiment was used only to evaluate whether the lag-alignment module can recover controlled delayed relationships; it was not used for model selection. For MSDS and SWaT, ground-truth lag annotations are unavailable. Therefore, we report representative learned-lag cases for proxy/contextual and endogenous-response pairs, and interpret them as temporally plausible associations rather than causal evidence.

4.4 Main Results

Table 2 compares EMS-uDTWAD with the MLP-enhanced baseline variants. EMS-uDTWAD obtains F1 scores of 0.936 on MSDS, 0.996 on the Linear Dataset, and 0.918 on SWaT. The margins over the strongest MLP-enhanced baseline are 2.0 percentage points on MSDS, 1.0 percentage point on the Linear Dataset, and 0.8 percentage points on SWaT. The gain is moderate rather than overwhelming, but it is consistent across the proxy-context, native-exogenous, and operational-context settings. The comparison is therefore conservative, since the baselines are also allowed to use the auxiliary-context fusion module. Compared with the standalone sDTW baseline, EMS-uDTWAD is also clearly better; a controlled replacement of DTW variants is reported in Section 4.5.3.

We further report how MLP fusion affects each baseline by comparing the original version with its MLP-enhanced counterpart. The results are reported in Table 3. The original baselines are evaluated without the auxiliary-context fusion module, while the "+MLP" variants use the same fusion setting described in Section 4.3. Both versions are tuned independently under the same validation protocol.

Table 2: F1-score comparison between EMS-uDTWAD and context-enhanced baseline methods on the three datasets.

Method	MSDS	Linear Dataset	SWaT
Informer+MLP [13]	0.901	0.969	0.828
TFAD+MLP [34]	0.810	0.714	0.628
Anomaly-Transformer+MLP [14]	0.791	0.957	<u>0.910</u>
FCVAE+MLP [9]	<u>0.916</u>	0.829	0.805
KAN-AD+MLP [35]	0.905	<u>0.986</u>	0.878
sDTW+MLP [28]	0.754	0.507	0.628
EMS-uDTWAD	**0.936**	**0.996**	**0.918**

Note: "+MLP" denotes the baseline version with the same auxiliary-context fusion module. F1 denotes the F1 score. Bold values indicate the best results, and underlined values indicate the second-best results.

Table 3: Effect of the MLP fusion module on baseline methods.

Method	MSDS			Linear Dataset			SWaT		
	Orig.	+MLP	Δ	Orig.	+MLP	Δ	Orig.	+MLP	Δ
Informer [13]	0.886	0.901	+0.015	0.956	0.969	+0.013	0.812	0.828	+0.016
TFAD [34]	0.798	0.810	+0.012	0.728	0.714	−0.014	0.641	0.628	−0.013
Anomaly-Transformer [14]	0.805	0.791	−0.014	0.942	0.957	+0.015	0.902	0.910	+0.008
FCVAE [9]	0.903	0.916	+0.013	0.842	0.829	−0.013	0.817	0.805	−0.012
KAN-AD [35]	0.893	0.905	+0.012	0.978	0.986	+0.008	0.862	0.878	+0.016
sDTW [28]	0.762	0.754	−0.008	0.519	0.507	−0.012	0.619	0.628	+0.009

Note: Orig. denotes the original baseline without the MLP fusion module. +MLP denotes the baseline version with the same auxiliary-context fusion module. Δ denotes the F1-score change after adding MLP fusion, computed as $F1_{+MLP} - F1_{Orig}$.

As shown in Table 3, the effect of MLP fusion varies across models and datasets. Informer and KAN-AD benefit from the MLP fusion on all three datasets, while TFAD, FCVAE, and sDTW show mixed or negative changes in some cases. Anomaly-Transformer also improves on the Linear Dataset and SWaT but decreases on MSDS. These results indicate that MLP fusion is not a uniformly beneficial add-on; its effect depends on the baseline architecture and the role of the auxiliary input. Together with the main comparison, these results show that the evaluation does not depend on a single baseline setting. EMS-uDTWAD is compared against context-enhanced baselines in the main table, while the original and MLP-enhanced versions are both reported here under the same validation protocol.

4.5 Model Analysis and Discussion

4.5.1 Ablation Study

To evaluate the role of each component in EMS-uDTWAD, several ablation variants are tested: (a) removing both the uDTW matching branch and the graph embedding module, and replacing them with a simple VAE; (b) using only the coarse-grained lag alignment and uDTW matching branch, denoted as the coarse-only variant; (c) using only the fine-grained graph embedding branch, denoted as the fine-only variant; (d) removing the multi-scale temporal separation and using a single window length for both branches, denoted as the single-scale framework; (e) reversing the module-scale assignment by applying the

lag alignment and uDTW matching branch to fine-grained windows and the graph embedding module to coarse-grained windows; (f) removing the auxiliary or contextual input and using only endogenous variables; (g) removing the MLP-based contextual fusion and using direct concatenation with linear projection; and (h) removing the explicit lag alignment mechanism while retaining synchronous concatenation of endogenous and contextual inputs. The ablation results are reported in Table 4.

Table 4: Ablation study on the MSDS, Linear Dataset, and SWaT.

Variant	MSDS F1	Linear F1	SWaT F1
w/o uDTW and GEM	0.564	0.536	0.517
Coarse-only (w/o GEM)	0.858	0.885	0.840
Fine-only (w/o uDTW)	0.731	0.754	0.762
Single-scale framework	<u>0.914</u>	<u>0.984</u>	<u>0.898</u>
Reversed scale-module pairing	0.872	0.927	0.878
w/o auxiliary/contextual input	0.902	0.941	0.879
w/o MLP contextual fusion	0.912	0.976	0.891
w/o lag alignment	0.910	0.972	0.885
EMS-uDTWAD	**0.936**	**0.996**	**0.918**

Note: GEM denotes the graph embedding module. "w/o auxiliary/contextual input" removes native exogenous variables, proxy auxiliary context, or operational contextual inputs according to the dataset. "w/o MLP contextual fusion" replaces the MLP-based contextual transformation with direct concatenation and linear projection. Bold values indicate the best F1 scores, and underlined values indicate the second-best F1 scores.

Table 4 shows that removing both the uDTW branch and the graph embedding module causes the largest performance loss. The coarse-only and fine-only variants are also weaker than EMS-uDTWAD, which supports the complementarity between coarse-grained lag-enhanced matching and fine-grained graph modeling. More importantly, the single-scale framework performs worse than EMS-uDTWAD even though both branches are retained. This result indicates that the gain comes not only from using two scoring branches, but also from assigning distinct temporal resolutions to them.

The auxiliary-input removal variant further confirms the role of exogenous or contextual inputs. The performance decrease is most visible on the Linear Dataset, where native exogenous variables are available. On MSDS, the decrease should be interpreted as the effect of proxy auxiliary context rather than evidence of native exogenous-variable modeling. On SWaT, the decrease indicates that actuator and control-state variables provide useful operational context.

Removing MLP contextual fusion also reduces performance, although the drop is smaller than that caused by removing the auxiliary input. This suggests that the auxiliary information is useful in itself, and that MLP-based contextual transformation helps the model exploit it more effectively. The lower scores of the reversed scale-module variant and the variant without lag alignment further support the current design, where lag-aware uDTW matching is applied at the coarse scale and graph embedding is applied at the fine scale.

4.5.2 Effect of Normal Reference-Library Size

Since the uDTW branch compares each test window with the normal reference library, the size of $\mathcal{R}_{\text{lag}}$ affects both detection performance and inference cost. To examine this effect, we vary the retained proportion of normal reference windows from 25% to 100%. The results are reported in Table 5.

Table 5: Effect of normal reference-library size on F1 score and inference time.

Library Ratio	MSDS		Linear Dataset		SWaT	
	F1	Time (ms)	F1	Time (ms)	F1	Time (ms)
25%	0.899	0.94	0.961	0.73	0.884	1.17
50%	0.918	1.46	0.985	1.12	0.904	1.86
75%	0.931	2.03	0.993	1.55	0.914	2.55
100%	**0.936**	2.47	**0.996**	1.96	**0.918**	3.18

Note: Time denotes average inference time per test window. The 100% setting corresponds to the reference-library sizes used in the main experiments, namely 768 for MSDS, 512 for the Linear Dataset, and 1024 for SWaT. Bold values indicate the best F1 scores.

The reference-library size mainly controls the balance between matching coverage and latency. Using only 25% or 50% of the library reduces inference time, but it also removes part of the normal-pattern diversity needed for nearest-neighbor matching. The improvement becomes smaller from 75% to 100%, indicating that a pruned library may be sufficient when latency is more important than the last fraction of F1 score. The full library is therefore used in the main experiments, while library pruning remains a practical option for deployment.

4.5.3 Effect of DTW Variants

To verify whether uDTW is necessary in the proposed pipeline, we further replace the uDTW-based matching distance in the coarse-grained branch with conventional DTW and sDTW. All variants use the same multi-scale windowing strategy, lag alignment module, graph embedding module, score fusion strategy, and threshold estimation. The only changed component is the sequence matching distance used in the coarse-grained branch. Therefore, this comparison isolates the contribution of uDTW-based uncertainty-aware matching from the other components of EMS-uDTWAD.

Table 6 isolates the effect of the matching distance in the coarse-grained branch. EMS-uDTWAD gives the highest F1 score on all three datasets, with gains of 0.026, 0.009, and 0.023 over EMS-DTWAD on MSDS, the Linear Dataset, and SWaT, respectively. EMS-sDTWAD improves over conventional DTW on MSDS and SWaT, but not on the Linear Dataset. This mixed result indicates that smoothing alone is not always sufficient. The additional uncertainty term in uDTW gives a more stable matching cost when the lag-enhanced windows contain local noise or distorted segments.

Table 6: Effect of DTW variants under the same multi-scale framework.

Variant	MSDS F1	Linear F1	SWaT F1
EMS-DTWAD	0.910	0.987	0.895
EMS-sDTWAD	0.927	0.985	0.903
EMS-uDTWAD	**0.936**	**0.996**	**0.918**

Note: All variants use the same multi-scale windowing, lag alignment, graph embedding module, score fusion strategy, and threshold estimation. Only the matching distance in the coarse-grained branch is changed. Bold values indicate the best F1 scores.

4.5.4 Contribution of Exogenous or Auxiliary Context

To quantify the contribution of exogenous or auxiliary contextual inputs, we compare EMS-uDTWAD with two controlled variants. The first variant uses only endogenous variables and removes all exogenous or

auxiliary contextual inputs. The second variant uses exogenous or auxiliary contextual inputs through direct synchronous concatenation, but removes the lag alignment step. The full EMS-uDTWAD further introduces lag-enhanced exogenous responses before uDTW-based matching. The results are reported in Table 7.

Table 7: Effect of exogenous or auxiliary contextual inputs.

Variant	MSDS F1	Linear F1	SWaT F1
Endogenous only	0.898	0.942	0.854
Endogenous + synchronous exogenous/context	0.910	0.972	0.885
Endogenous + lag-enhanced exogenous/context	**0.936**	**0.996**	**0.918**

Note: For MSDS, the auxiliary input is the load-fluctuation proxy rather than a native external driver. For SWaT, actuator and control-state variables are used as operational contextual inputs. Bold values indicate the best F1 scores.

Removing the auxiliary input consistently reduces F1. The drop is 0.038 on MSDS, 0.054 on the Linear Dataset, and 0.064 on SWaT. Direct synchronous concatenation recovers part of this loss, but it remains weaker than the full lag-enhanced model. The difference between synchronous concatenation and EMS-uDTWAD is 0.026 on MSDS, 0.024 on the Linear Dataset, and 0.033 on SWaT. These gaps support the role of the lag-alignment step, while the MSDS result should be interpreted only as evidence for proxy auxiliary context rather than for a genuine external driver.

4.5.5 Robustness under Missing Exogenous or Contextual Inputs

In practical scenarios, contextual inputs may contain missing values or noise because of acquisition faults or communication instability. We therefore evaluate EMS-uDTWAD by masking only the auxiliary input of each dataset: the load-fluctuation proxy in MSDS, native exogenous variables in the Linear Dataset, and actuator/control-state variables in SWaT. The endogenous variables remain unchanged. Two perturbation strategies are considered: **Zeros**, where masked values are replaced with 0, and **Random**, where masked values are replaced with random values sampled from a normal distribution.

The masking ratios are set to 10%, 25%, and 50%. For each ratio, the reported F1 score is averaged over the zero-value masking and random-noise masking settings. As shown in Fig. 4, EMS-uDTWAD maintains stable performance when auxiliary inputs are partially missing or disturbed, suggesting reasonable robustness under the evaluated settings.

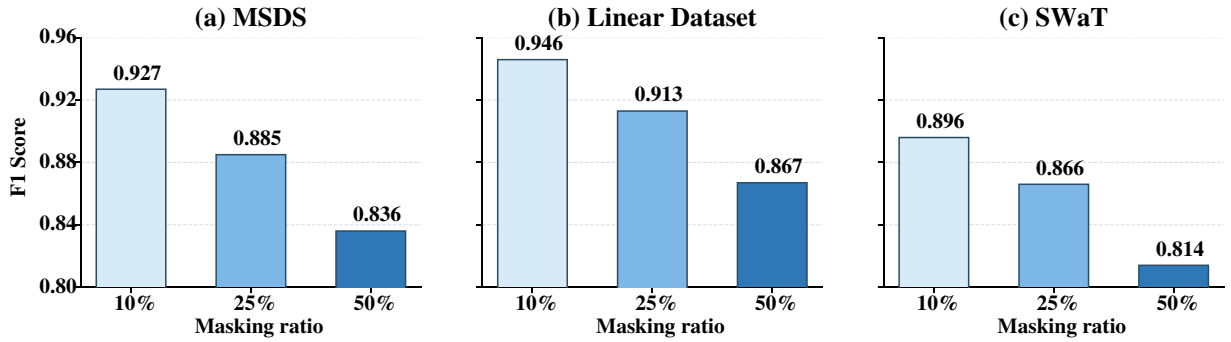

Figure 4: Robustness analysis under different missing ratios of exogenous or contextual inputs: (**a**) MSDS with the load-fluctuation proxy masked; (**b**) Linear Dataset with native exogenous variables masked; (**c**) SWaT with actuator/control-state variables masked.

4.5.6 Visualization Analysis with Proxy Auxiliary Context

To examine the effect of proxy auxiliary context on anomaly detection performance, this study visualizes the anomaly detection results on the MSDS dataset. Specifically, CPU, Memory, and Disk are selected as representative endogenous variables, while the constructed load-fluctuation proxy is used as auxiliary context to characterize short-term system-level fluctuation patterns. As shown in Fig. 5, panel (a) presents the load-fluctuation proxy. Panels (b), (c), and (d) show the anomaly detection results of CPU, Memory, and Disk when proxy auxiliary context is unavailable, respectively. Panels (e), (f), and (g) show the corresponding anomaly detection results when proxy auxiliary context is available. The red shaded regions indicate anomaly intervals.

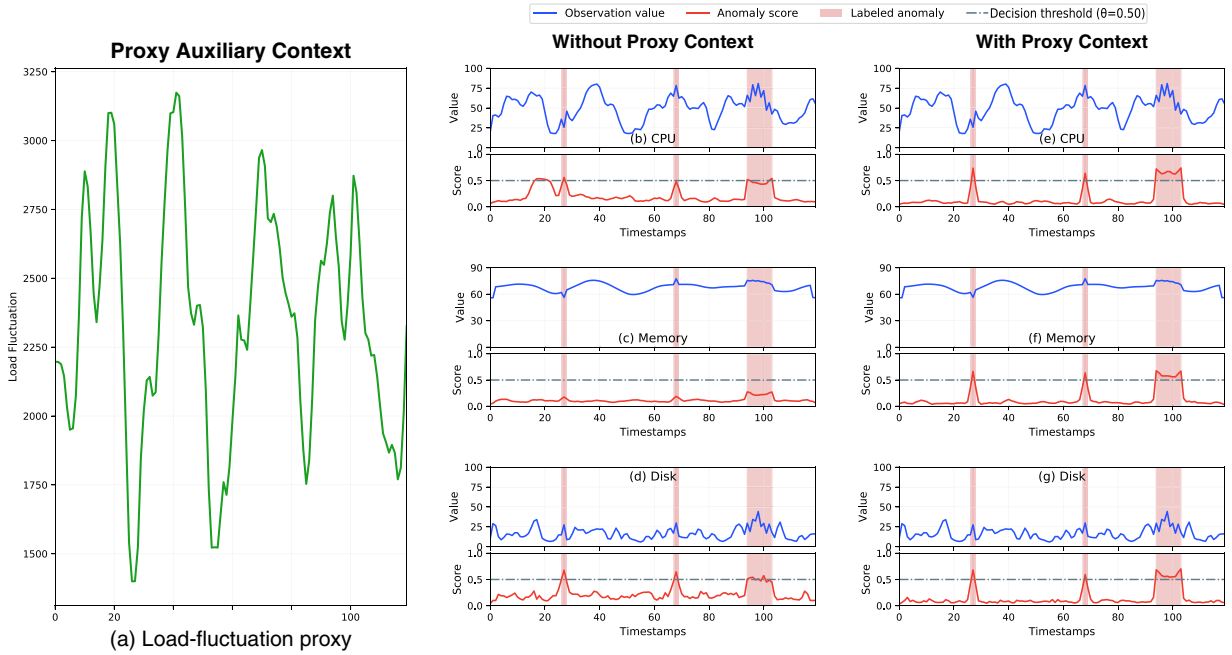

Figure 5: Visualization comparison of anomaly detection results with and without proxy auxiliary context on the MSDS dataset: (**a**) Load-fluctuation proxy; (**b**) CPU without proxy auxiliary context; (**c**) Memory without proxy auxiliary context; (**d**) Disk without proxy auxiliary context; (**e**) CPU with proxy auxiliary context; (**f**) Memory with proxy auxiliary context; (**g**) Disk with proxy auxiliary context. Red shaded regions indicate anomaly intervals.

The results show that the model can still respond to anomaly intervals without proxy auxiliary context. However, the score peaks are less prominent, and some anomaly boundaries remain ambiguous. In contrast, after introducing proxy auxiliary context, the anomaly score curves of different variables exhibit sharper and more concentrated variations around anomaly intervals, with more pronounced score peaks. This result suggests that the load-fluctuation proxy provides useful auxiliary context on MSDS. It should not be interpreted as evidence that MSDS contains native exogenous variables. This visualization is intended to illustrate the effect of proxy-context availability on anomaly-score localization, whereas the quantitative contribution of lag-aware modeling is evaluated through the ablation study in Table 7.

4.5.7 Parameter Sensitivity Analysis

To further investigate the influence of key parameters on model performance, this study analyzes the effect of different window sizes. Window-size sensitivity is evaluated on the MSDS and Linear Dataset by varying the coarse-grained and fine-grained window sizes, with the F1 score used as the evaluation metric.

These two datasets respectively represent a real-world proxy auxiliary-context setting and a controlled native-exogenous-variable setting. The sensitivity of the fusion coefficient α is further evaluated on all three datasets, including SWaT. Fig. 6 reports representative window-size sensitivity results on MSDS and the Linear Dataset, where the scale coefficient λ is set to 2 and 4 to evaluate different fine- and coarse-grained window-size combinations.

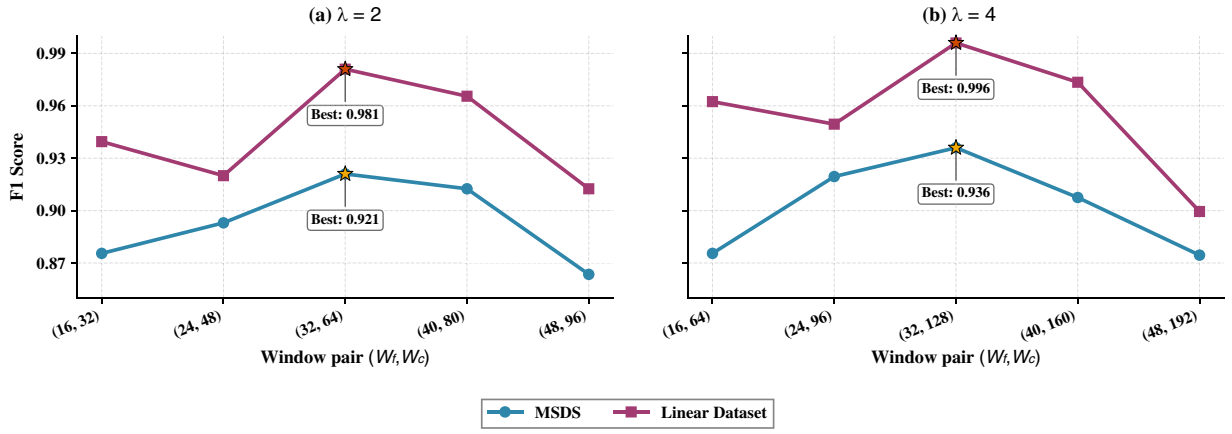

Figure 6: Parameter sensitivity analysis of multi-scale window-size combinations on MSDS and the Linear Dataset under different values of the scale coefficient λ: (**a**) $\lambda = 2$; (**b**) $\lambda = 4$. The curves report the F1 scores obtained with different fine- and coarse-grained window-size pairs (W_f, W_c), where W_f and W_c denote the fine-grained and coarse-grained window sizes, respectively.

The horizontal axis in Fig. 6 represents the combinations of fine-grained and coarse-grained windows, while the vertical axis denotes the best F1 score. The experimental results show that the model achieves the best performance when $\lambda = 4$, corresponding to $W_{\text{fine}} = 32$ and $W_{\text{coarse}} = 128$. This setting is consistent with the window configuration used in the main experiments. Under this configuration, the best F1 scores reach 0.936 on the MSDS dataset and 0.996 on the Linear Dataset. When the window sizes are either too small or too large, the F1 score decreases to varying degrees. This result indicates that an appropriate multi-scale window setting helps the model capture both fine-grained local anomalies and coarse-grained segment-level patterns, whereas unsuitable window scales may weaken detection performance. The influence of the fusion coefficient α is also examined. In this experiment, α is varied from 0 to 1 with a step size of 0.1. The two endpoints correspond to using only the graph embedding branch and only the uDTW branch, respectively. The results are reported in Table 8.

Table 8: Sensitivity analysis of the fusion coefficient α on the MSDS, Linear Dataset, and SWaT.

α	MSDS F1	Linear F1	SWaT F1
0.0	0.731	0.754	0.762
0.1	0.798	0.781	0.801
0.2	0.856	0.842	0.846
0.3	0.891	0.914	0.874
0.4	0.917	0.974	0.894
0.5	0.932	0.986	0.907
0.6	0.923	**0.996**	**0.918**
0.7	**0.936**	0.981	0.913

(Continued)

Table 8 (continued)

α	MSDS F1	Linear F1	SWaT F1
0.8	0.902	0.949	0.882
0.9	0.886	0.903	0.856
1.0	0.858	0.885	0.840

Note: Bold values indicate the best F1 scores in each dataset. The endpoints $\alpha = 0$ and $\alpha = 1$ are included as boundary references, corresponding to graph-only and uDTW-only score fusion, respectively.

The results show that the fused setting performs better than the single-branch boundary cases in most cases. The best F1 score is obtained when $\alpha = 0.7$ on MSDS and when $\alpha = 0.6$ on the Linear Dataset and SWaT. This indicates that the uDTW branch contributes slightly more to the fused score, while the graph embedding branch still provides complementary local structural information. These results are consistent with the validation-based parameter selection used in the main experiments.

4.5.8 Analysis of Learned Lag Values

To further examine whether the lag-alignment module learns meaningful delayed relationships, we record the soft lag estimate $\hat{\tau}_{n,d}^{(c)}$ defined in Section 3.4. For each endogenous–contextual channel pair, the average learned lag over normal validation windows is computed as

$$\bar{\tau}_{n,d} = \frac{1}{|\mathcal{C}_{\text{val}}|} \sum_{c \in \mathcal{C}_{\text{val}}} \hat{\tau}_{n,d}^{(c)}, \tag{43}$$

where C_{val} denotes the set of normal validation windows.

As shown in Table 9, the learned lag values on the Linear Dataset are close to the controlled delayed relationships. The average absolute lag error on the controlled Linear Dataset pairs remains within one time step, suggesting that the lag-alignment module can capture delayed exogenous-to-endogenous responses under a setting where the lag structure is known. For MSDS and SWaT, no ground-truth lag annotations are provided. Therefore, the learned lags are interpreted only as temporally plausible associations between proxy or operational context and endogenous responses, rather than as causal evidence.

Table 9: Analysis of learned lag values on the three datasets.

Dataset	Channel Pair	True Lag	Learned Lag	Interpretation
Linear Dataset	$v^{(1)} \to x^{(1)}$	2	2.14	Close to the controlled delay.
Linear Dataset	$v^{(2)} \to x^{(2)}$	4	3.82	Follows the assigned delayed relationship.
Linear Dataset	$v^{(3)} \to x^{(3)}$	6	6.27	Small deviation due to soft lag weighting.
MSDS	Load-fluctuation proxy → CPU utilization	N/A	2.63	Proxy-context changes precede the CPU response within a short lag range.

(Continued)

Table 9 (continued)

Dataset	Channel Pair	True Lag	Learned Lag	Interpretation
SWaT	Actuator/control state → flow or level measurement	N/A	3.18	Consistent with a delayed operational response after control-state changes.

Note: N/A = not available, indicating that no ground-truth lag annotation is available for the corresponding dataset. For the Linear Dataset, the true lag denotes the controlled delayed relationship used for diagnostic analysis. For MSDS and SWaT, ground-truth lag annotations are unavailable; therefore, the learned lags are interpreted as temporally plausible associations rather than causal evidence.

4.5.9 Computational Complexity and Inference Efficiency

We further analyze the computational cost of EMS-uDTWAD because the uDTW branch compares each test window with the normal reference library. Let C_{te} denote the number of test windows, N and D denote the numbers of endogenous and exogenous or contextual variables, $M = N + D$, $K_{\max}$ denote the maximum lag, W_{coarse} denote the coarse-grained window length, p' denote the latent sequence length after convolution, d denote the latent feature dimension, and $|\mathcal{R}_{\text{lag}}|$ denote the number of windows in the lag-enhanced normal reference library.

The lag-alignment module evaluates candidate lags between endogenous and contextual channels and constructs the lag-compensated representation. Its complexity is

$$O(C_{\text{te}} N D (K_{\max} + 1) W_{\text{coarse}}) . \tag{44}$$

The graph branch mainly includes adjacency construction and graph reconstruction. Its dominant cost is

$$O\left(C_{\text{te}} M^2 d_g\right), \tag{45}$$

where d_g is the graph embedding dimension. The dominant inference cost comes from the uDTW matching branch. For each test window, the uDTW distance is computed against all reference windows, and each pairwise matching has a cost of $O(p'^2 d)$. Therefore, the uDTW branch has complexity

$$O(C_{\text{te}} |\mathcal{R}_{\text{lag}}| p'^2 d). \tag{46}$$

The overall inference complexity can be summarized as

$$O\left(C_{\text{te}} N D (K_{\max} + 1) W_{\text{coarse}} + C_{\text{te}} M^2 d_g + C_{\text{te}} |\mathcal{R}_{\text{lag}}| p'^2 d\right). \tag{47}$$

This formulation shows that the reference-library size and the latent sequence length are the main factors controlling inference cost.

Table 10 reports the average inference time per test window for representative forecasting-based, Transformer-based, reconstruction-based, and DTW-based methods. EMS-uDTWAD is slower than standard neural-network baselines due to lag alignment and reference-library matching, but remains faster than the standalone sDTW baseline under the same environment.

Table 10: Inference efficiency comparison on the three datasets.

Method	MSDS	Linear Dataset	SWaT
Informer [13]	0.81	0.63	1.02
Anomaly-Transformer [14]	1.26	0.97	1.54
FCVAE [9]	0.74	0.58	0.91
sDTW [28]	6.83	5.42	8.11
EMS-uDTWAD	2.47	1.96	3.18

Note: Values denote average inference time per test window in milliseconds. Inference time is measured after model loading and excludes data preprocessing, file I/O, and visualization.

Memory cost is mainly determined by model parameters, cached reference-library representations, and the temporary pairwise distance matrix used in uDTW. During inference, the cached lag-enhanced reference representation requires $O(|\mathcal{R}_{\text{lag}}|p'd)$ memory, the cached graph-reference representation requires $O(|\mathcal{R}_{\text{G}}|M^2)$ memory, and the temporary uDTW distance matrix requires $O(p'^2)$ memory for each pairwise comparison. Thus, the main memory cost can be written as

$$O\left(|\Theta| + |\mathcal{R}_{\text{lag}}|p'd + |\mathcal{R}_{\text{G}}|M^2 + p'^2\right), \tag{48}$$

where $|\Theta|$ denotes the number of model parameters and $|\mathcal{R}_{\text{G}}|$ denotes the size of the graph-reference library. The measured memory costs are reported in Table 11.

Table 11: Memory cost of EMS-uDTWAD during inference.

Dataset	$\lvert\mathbf{R}_{\text{lag}}\rvert$	Model Parameters (MB)	Cached Reference Library (MB)	Peak GPU Memory (MB)
MSDS	768	3.92	8.6	421
Linear Dataset	512	3.88	5.4	368
SWaT	1024	4.35	22.7	612

Note: The cached reference library includes lag-enhanced uDTW features and graph-reference representations. Peak GPU memory is measured during test-window inference with the same batch size used in the main experiments.

The main overhead of EMS-uDTWAD comes from reference-library matching rather than from the neural components. This behavior is expected from the complexity term $O(C_{\text{te}}|\mathcal{R}_{\text{lag}}|p'^2d)$. In the reported implementation, the per-window inference time remains at the millisecond level after reference features are cached. For stricter real-time deployment, the library size, latent sequence length, and update frequency should be reduced jointly instead of recomputing the full reference set at every inference step.

Edge deployment requires additional engineering beyond the current experimental setting. Edge computing can place inference closer to the monitored device and reduce communication delay [39]. Weight quantization and network pruning may further reduce memory traffic and neural-network inference cost [40], while transfer learning may reduce retraining cost when the operating condition changes [2]. These techniques were not applied here; the reported time and memory therefore correspond to the uncompressed model with cached reference features.

5 Conclusion and Future Work

This work clarifies the role of uDTW in EMS-uDTWAD by separating lag modeling from sequence matching. The coarse-grained branch first estimates delayed contextual responses and constructs lag-enhanced windows, while uDTW is used only to measure the deviation between lag-enhanced test windows and the normal reference library. The fine-grained graph branch models local structural dependencies through a learned adjacency representation. This design avoids treating uDTW as a direct estimator of exogenous-to-endogenous lags and makes the function of each module more explicit.

Experiments on MSDS, the Linear Dataset, and SWaT show consistent F1 improvements over the compared baselines under proxy-context, native-exogenous, and operational-context settings. The ablation results further show that removing auxiliary inputs, removing lag alignment, using a single scale, or reversing the scale-module assignment all degrades performance. The learned-lag analysis provides additional evidence that the lag-alignment module can recover controlled delayed relationships on the Linear Dataset, while the MSDS and SWaT examples show temporally plausible contextual-response patterns. The DTW-variant comparison also supports the use of uDTW in the coarse-grained matching branch. The computational analysis confirms that the main overhead comes from matching test windows with the normal reference library, but cached reference features keep the reported per-window inference time at the millisecond level.

One limitation is that public real-world datasets with officially annotated native exogenous variables remain limited. Therefore, the MSDS results should be interpreted as proxy auxiliary-context evidence, SWaT as operational-context evidence, and the Linear Dataset as controlled native-exogenous evidence. Future work will focus on real industrial datasets with clearer external drivers and on deployment-oriented acceleration, including reference-library pruning, model compression, and adaptation under changing operating conditions.

Acknowledgement: Not applicable.

Funding Statement: This work was supported in part by the Major Science and Technology Special Project of Qinghai Office of Science and Technology, China, under Grant No. 2024-GX-A3, and in part by the Kunlun Talent High-End Innovation and Entrepreneurship Talent Project in Qinghai Province, China.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Yuanzhao Shang, Xin Liu; data collection: Yuanzhao Shang; analysis and interpretation of results: Yuanzhao Shang, Xin Liu, Fengbiao Zan; draft manuscript preparation: Yuanzhao Shang; supervision: Xin Liu. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data supporting the findings of this study are available from public repositories or from the corresponding dataset providers. The MSDS dataset is publicly available on Zenodo at https://doi.org/10.5281/zenodo.3484800. The source code and data-generation settings for the Linear Dataset are available at https://github.com/i6092467/GVAR. The synthetic Linear Dataset used in this study was generated and processed following the experimental settings described in this article. The SWaT dataset is available upon request from the iTrust, Centre for Research in Cyber Security, Singapore University of Technology and Design, subject to the dataset access agreement. Additional implementation details are available from the corresponding author upon reasonable request.

Ethics Approval: Not applicable. This study did not involve human participants or animals.

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

References

1. Wen Q, Yang L, Zhou T, Sun L. Robust time series analysis and applications: an industrial perspective. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining; 2022 Aug 14–18; Washington, DC, USA. New York, NY, USA: Association for Computing Machinery; 2022. p. 4836–7. doi:10.1145/3534678.3542612.
2. Yan P, Abdulkadir A, Luley PP, Rosenthal M, Schatte GA, Grewe BF, et al. A comprehensive survey of deep transfer learning for anomaly detection in industrial time series: methods, applications, and directions. IEEE Access. 2024;12(1):3768–89. doi:10.1109/ACCESS.2023.3349132.
3. Kieu T, Yang B, Jensen CS. Outlier detection for multidimensional time series using deep neural networks. In: Proceedings of the 2018 19th IEEE International Conference on Mobile Data Management (MDM); 2018 Jun 25–28; Aalborg, Denmark. Piscataway, NJ, USA: IEEE; 2018. p. 125–34. doi:10.1109/MDM.2018.00029.
4. Shentu Q, Li B, Zhao K, Shu Y, Rao Z, Pan L, et al. Towards a general time series anomaly detector with adaptive bottlenecks and dual adversarial decoders. arXiv:2405.15273. 2025.
5. Yeh CCM, Zhu Y, Ulanova L, Begum N, Ding Y, Dau HA, et al. Matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In: Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM); 2016 Dec 12–15; Barcelona, Spain. Piscataway, NJ, USA: IEEE; 2016. p. 1317–22. doi:10.1109/ICDM.2016.0179.
6. Vallis O, Hochenbaum J, Kejariwal A. A novel technique for long-term anomaly detection in the cloud. In: Proceedings of the 6th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 14); 2014 Jun 17–18; Philadelphia, PA, USA. Berkeley, CA, USA: USENIX Association; 2014.
7. Liu FT, Ting KM, Zhou ZH. Isolation forest. In: Proceedings of the 2008 8th IEEE International Conference on Data Mining (ICDM); 2008 Dec 15–19; Pisa, Italy. Piscataway, NJ, USA: IEEE; 2008. p. 413–22. doi:10.1109/ICDM.2008.17.
8. Ramaswamy S, Rastogi R, Shim K. Efficient algorithms for mining outliers from large data sets. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data; 2000 May 16–18; Dallas, TX, USA. New York, NY, USA: Association for Computing Machinery; 2000. p. 427–38. doi:10.1145/342009.335437.
9. Wang Z, Pei C, Ma M, Wang X, Li Z, Pei D, et al. Revisiting VAE for unsupervised time series anomaly detection: a frequency perspective. In: Proceedings of the ACM Web Conference 2024; 2024 May 13–17; Singapore. New York, NY, USA: Association for Computing Machinery; 2024. p. 3096–105. doi:10.1145/3589334.3645710.
10. Lai KH, Zha D, Xu J, Zhao Y, Wang G, Hu X. Revisiting time series outlier detection: definitions and benchmarks. In: Proceedings of the 35th Conference on Neural Information Processing Systems, Datasets and Benchmarks Track; 2021 Dec 6–14; Virtual.
11. Kim H, Kim S, Min S, Lee B. Contrastive time-series anomaly detection. IEEE Trans Knowl Data Eng. 2024;36(10):5053–65. doi:10.1109/TKDE.2023.3335317.
12. Yang Y, Zhang C, Zhou T, Wen Q, Sun L. DCdetector: dual attention contrastive representation learning for time series anomaly detection. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining; 2023 Aug 6–10; Long Beach, CA, USA. New York, NY, USA: Association for Computing Machinery; 2023. p. 3033–45. doi:10.1145/3580305.3599295.
13. Zhou H, Zhang S, Peng J, Zhang S, Li J, Xiong H, et al. Informer: beyond efficient transformer for long sequence time-series forecasting. Proc AAAI Conf Artif Intell. 2021;35(12):11106–15. doi:10.1609/aaai.v35i12.17325.
14. Xu J, Wu H, Wang J, Long M. Anomaly transformer: time series anomaly detection with association discrepancy. arXiv:2110.02642. 2022.
15. Guo R, Li A, Liu H. An adversarial attack detection method based on bidirectional consistency discrimination for deep learning-based soft sensors. In: Proceedings of the 2025 CAA Symposium on Fault Detection, Supervision, and Safety for Technical Processes (SAFEPROCESS); 2025 Aug 22–24; Urumqi, China. Piscataway, NJ, USA: IEEE; 2025. p. 1–6. doi:10.1109/SAFEPROCESS67117.2025.11268032.
16. Zhang W, Luo C. Decomposition-based multi-scale transformer framework for time series anomaly detection. Neural Netw. 2025;187(2):107399. doi:10.1016/j.neunet.2025.107399.

17. Qiu X, Zhu Y, Li Z, Cheng H, Wu X, Guo C, et al. DAG: a dual causal network for time series forecasting with exogenous variables. arXiv:2509.14933. 2025.
18. Wang Y, Wu H, Dong J, Qin G, Zhang H, Liu Y, et al. TimeXer: empowering transformers for time series forecasting with exogenous variables. Adv Neural Inf Process Syst. 2024;37:469–98. doi:10.52202/079017-0015.
19. Cuturi M. Fast global alignment kernels. In: Proceedings of the 28th International Conference on Machine Learning (ICML-11); 2011 Jun 28–Jul 2; Bellevue, WA, USA. Madison, WI, USA: Omnipress; 2011. p. 929–36.
20. Wang L, Koniusz P. Uncertainty-DTW for time series and sequences. In: Proceedings of the European Conference on Computer Vision (ECCV); 2022 Oct 23–27; Cham, Switzerland: Springer Nature Switzerland; 2022. p. 176–95.
21. Zhang L, Li Q, Yang Y, Chen J, Zeng R, Lyu C, et al. Contextual and seasonal LSTMs for time series anomaly detection. arXiv:2602.09690. 2026.
22. Wu X, Qiu X, Li Z, Wang Y, Hu J, Guo C, et al. CATCH: channel-aware multivariate time series anomaly detection via frequency patching. arXiv:2410.12261. 2025.
23. Goswami M, Szafer K, Choudhry A, Cai Y, Li S, Dubrawski A. MOMENT: a family of open time-series foundation models. In: Proceedings of the 41st International Conference on Machine Learning; 2024 Jul 21–27; Vienna Austria. Cambridge, MA, USA: PMLR; 2024. p. 16115–52.
24. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. Adv Neural Inf Process Syst. 2017;30:5998–6008.
25. Li Z, Qiu X, Zhu Y, Wu X, Hu J, Guo C, et al. GCGNet: graph-consistent generative network for time series forecasting with exogenous variables. arXiv:2603.08032. 2026. doi:10.48550/arXiv.2603.08032.
26. Gubbi J, Buyya R, Marusic S, Palaniswami M. Internet of Things (IoT): a vision, architectural elements, and future directions. Future Gener Comput Syst. 2013;29(7):1645–60. doi:10.1016/j.future.2013.01.010.
27. Konatham BR, Simra T, Ghimire A, Amsaad F, Ibrahem MI, Jhanjhi NZ. MI-assisted security for anomaly detection in industrial IoT (IIoT) applications. In: Proceedings of the 2023 Second International Conference on Smart Technologies for Smart Nation (SmartTechCon); 2023 Aug 18–19; Singapore. Piscataway, NJ, USA: IEEE; 2023. p. 1120–7. doi:10.1109/SmartTechCon57526.2023.10391331.
28. Cuturi M, Blondel M. Soft-DTW: a differentiable loss function for time-series. In: Proceedings of the 34th International Conference on Machine Learning (ICML); 2017 Aug 6–11; Sydney, Australia. Cambridge, MA, USA: PMLR; 2017. p. 894–903.
29. Ulyanov D, Vedaldi A, Lempitsky V. Instance normalization: the missing ingredient for fast stylization. arXiv:1607.08022. 2016.
30. Simonovsky M, Komodakis N. GraphVAE: towards generation of small graphs using variational autoencoders. In: Proceedings of the International Conference on Artificial Neural Networks (ICANN); 2018 Oct 4–7; Rhodes, Greece. Cham, Switzerland: Springer International Publishing; 2018. p. 412–22.
31. Nedelkoski S, Bogatinovski J, Mandapati AK, Becker S, Cardoso J, Kao O. Multi-source distributed system data for AI-powered analytics. Zenodo. 2019. doi:10.5281/zenodo.3484800.
32. Marcinkevičs R, Vogt JE. Interpretable models for Granger causality using self-explaining neural networks. arXiv:2101.07600. 2021.
33. Mathur AP, Tippenhauer NO. SWaT: a water treatment testbed for research and training on ICS security. In: Proceedings of the 2016 International Workshop on Cyber-Physical Systems for Smart Water Networks (CySWater); 2016 Apr 11; Vienna, Austria. Piscataway, NJ, USA: IEEE; 2016. p. 31–6. doi:10.1109/CySWater.2016.7469060.
34. Zhang C, Zhou T, Wen Q, Sun L. TFAD: a decomposition time series anomaly detection architecture with time-frequency analysis. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM); 2022 Oct 17–21; Atlanta, GA, USA. New York, NY, USA: Association for Computing Machinery; 2022. p. 2497–507. doi:10.1145/3511808.3557470.
35. Zhou Q, Pei C, Sun F, Jing H, Gao Z, Zhang H, et al. KAN-AD: time series anomaly detection with Kolmogorov–Arnold networks. In: Proceedings of the Forty-Second International Conference on Machine Learning; 2025 Jul 13–19; Vancouver, BC, Canada. Cambridge, MA, USA: PMLR; 2025. p. 79136–49.

36. Sarfraz MS, Chen MY, Layer L, Peng K, Koulakis M. Position: quo vadis, unsupervised time series anomaly detection? In: Proceedings of the 41st International Conference on Machine Learning; 2024 Jul 21–27; Vienna Austria. Cambridge, MA, USA: PMLR; 2024. p. 43461–76.
37. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: an imperative style, high-performance deep learning library. Adv Neural Inf Process Syst. 2019;32:8024–35.
38. Kingma DP, Ba J. Adam: a method for stochastic optimization. arXiv:1412.6980. 2015.
39. Shi W, Cao J, Zhang Q, Li Y, Xu L. Edge computing: vision and challenges. IEEE Internet Things J. 2016;3(5):637–46. doi:10.1109/JIOT.2016.2579198.
40. Deng L, Li G, Han S, Shi L, Xie Y. Model compression and hardware acceleration for neural networks: a comprehensive survey. Proc IEEE. 2020;108(4):485–532. doi:10.1109/JPROC.2020.2976475.