LEARN HYBRID PROTOTYPES FOR MULTIVARIATE TIME SERIES ANOMALY DETECTION

Anonymous authors
Paper under double-blind review

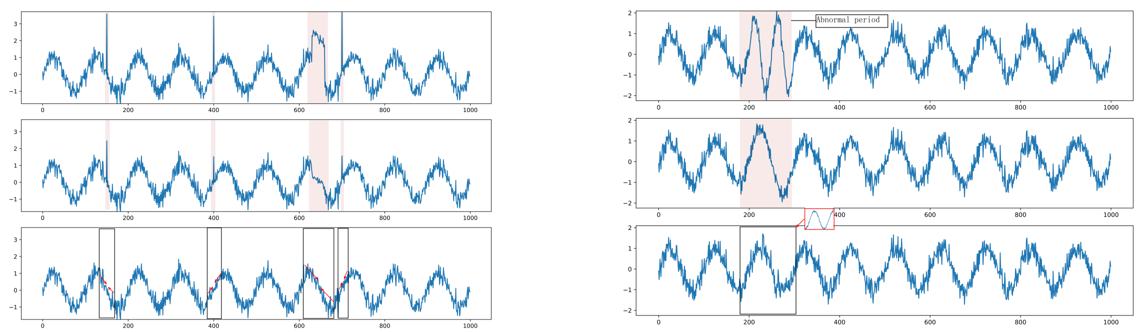
ABSTRACT

In multivariate time series anomaly detection (MTSAD), reconstruction-based models reconstruct testing series with learned knowledge of only normal series and identify anomalies with higher reconstruction errors. In practice, over-generalization often occurs with unexpectedly well reconstruction of anomalies. Although memory banks are employed by reconstruction-based models to fight against over-generalization, these models are only efficient to detect point anomalies since they learn normal prototypes from time points, leaving contextual anomalies and periodical anomalies to be discovered. To settle this problem, this paper propose a hybrid prototypes learning model for MTSAD based on reconstruction, named as H-PAD. First, normal prototypes are learned from different sizes of patches for time series to discover short-term anomalies. These prototypes in different sizes are integrated together to reconstruct query series so that any anomalies would be smoothed off and high reconstruction errors are produced. Furthermore, period prototypes are learned to discover periodical anomalies. One period prototype is memorized for one variable of query series. Finally, extensive experiments on five benchmark datasets show the effectiveness of H-PAD with state-of-the-art performance.

1 INTRODUCTION

Anomaly detection in multivariate time series is a common but important issue in many fields such as equipment monitoring, healthcare systems and aerospace engineering. Since labeling is time-consuming and labor-intensive, multivariate time series anomaly detection (MTSAD) is regarded as an unsupervised learning task(Eldele et al., 2021). Generally speaking, MTSAD learns knowledge directly from a set of normal data, and detects anomalies with learned normal knowledge.

The most popular MTSAD methods are developed based on reconstruction of time series. In the training phase, a set of time series of only normal points are featured in latent space by an encoder. Following with a decoder, the training set of time series are expected to reconstructed with least losses. In the inference phase, the trained encoder and decoder try to reconstruct a new time series and anomaly points would be discovered. However, the best reconstruction of training time series raises the problem of over-generalization for testing time series, shown as in Figure 1(a). That is, not only normal points in time series are reconstructed excellently, but abnormal points are also reconstructed very well. As a result, abnormal points can not be discovered because they can not be identified with high reconstruction errors no longer. To fight against over-generalization, MEMTO employs a memory bank of normal point prototypes to help reconstruct time series (Song et al., 2024). However, contextual information should be seriously considered for learning time series since each point is closely related with its neighbours. Moreover, the absence of periodicity in MTSAD lead to the fact that it is difficult to identify long-term anomalies (e.g. period anomalies in Figure 1(b)) only with point prototypes.



(a) Points and short-term anomalies (b) Period anomaly

Figure 1: Illustrations of over-generalization. The top time series is the testing time series including anomalies (highlighted in pink). The middle time series is its reconstructed series due to over-generalization that anomalies are reconstructed as well as normal ones. After all, the testing series is expected to be reconstructed as the bottom series, immune to anomaly influence.

This paper proposes an MTSAD model based on learning hybrid prototypes (H-PAD) which consist of patch prototypes in different scales and periodical prototypes (shown as in Figure 2). First of all, H-PAD is designed to learn memory prototypes of patches in different scales, instead of prototypes of time points. With patch prototypes, contextual information in time series is taken into consideration for future reconstruction. Occasional point anomalies can not be reconstructed well with their contextual information, further preventing the occurrence of over-generalization. Moreover, taking periodicity of time series into account, H-PAD also learns and memorizes period prototypes for time series in multiple variables, one period prototype for one variable. It enables the model to identify long-term anomalies because period prototypes can help to reconstruct the testing time series as a normal series which deviates greatly from input series. Experimental results on five benchmarks illustrate the effectiveness of H-PAD.

The contributions of this paper are as follows:

- We propose a novel framework to learn hybrid prototypes for multivariate time series anomaly detection. Patch prototypes involves local information and period prototypes contains global information. The combination of patch and period prototypes can well discover point anomalies and long-term anomalies as well as period anomalies. .
- Patch prototypes can utilize information of different patch sizes. With normal patch prototypes of different patch sizes, both normal and abnormal sequences are reconstructed to normal sequences such that the high reconstruction errors for abnormal sequences help the model to detect anomalies.
- The differences between normal prototypes in different patch and points and the differences between periods are taken into account in the anomaly score, which can more accurately identify anomalies.

2 RELATED WORKS

Early anomaly detection methods are generally some statistical methods and classical machine learning methods. Common statistical methods include the use of moving averages, exponential smoothing and the autoregressive integrated moving average model. Machine learning methods include classification-based, density-based and clustering-based methods. Classification-based methods such as decision trees (Liu et al.,

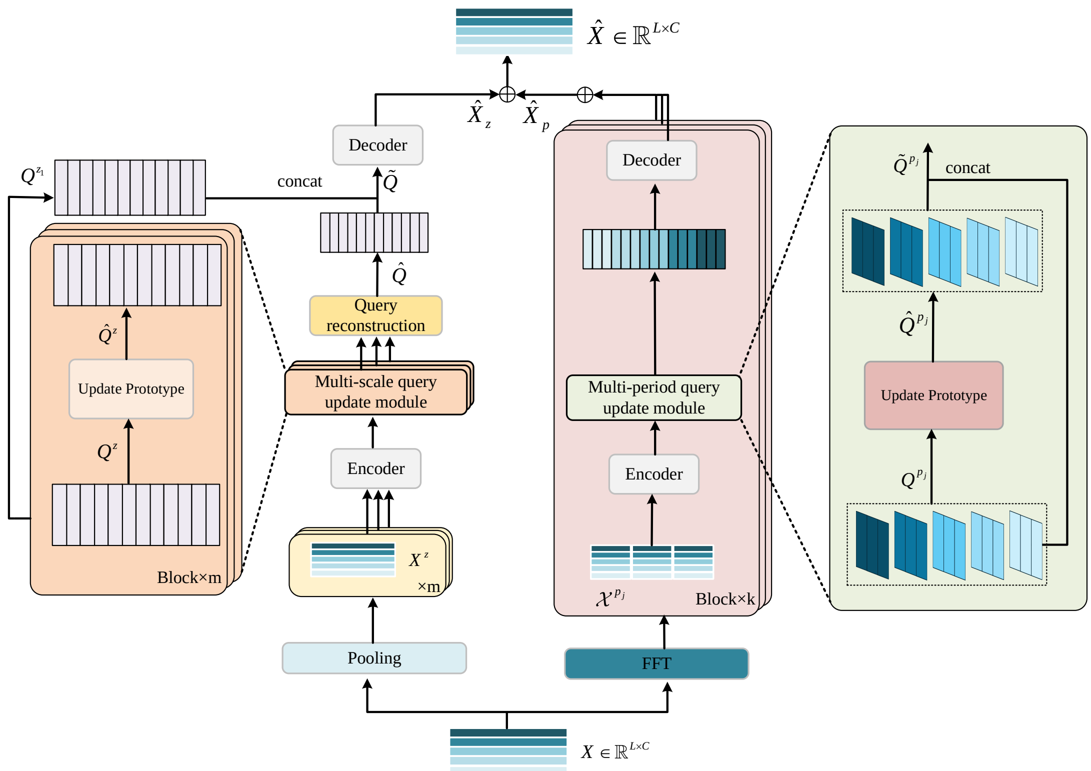


Figure 2: The overall architecture of the H-PAD model.

2008), support vector machines, one-class SVM (Schölkopf et al., 2001) and support vector data description (Tax & Duin, 2004) were widely used in the field of anomaly detection in the early days. Density-based methods such as local outlier factor (Breunig et al., 2000) and connectivity outlier factor (Tang et al., 2002) calculate local density and local connectivity respectively to determine outliers. In recent years, methods combining density estimation with deep learning, such as DAGMM (Zong et al., 2018) and MPPCACD (Yairi et al., 2017), have also been proposed. Common clustering methods include k-means (Kant & Mahajan, 2019), THOC (Shen et al., 2020), and ITAD (Shin et al., 2020).

Traditional methods have difficulty capturing the complex nonlinear relationships and temporal dependencies in time series data, which limits the performance of the model. Therefore, deep learning methods have become the mainstream method for anomaly detection, which is more effective in identifying anomalies in time series data. The most commonly used method is reconstruction-based. Early methods include LSTM-based encoder-decoder models and LSTM-VAE models (Park et al., 2018). Later, OmniAnomaly (Su et al., 2019) and InterFusion (Li et al., 2021) further extended the LSTM-VAE model. With the deepening of research on reconstruction models, reconstruction models combined with generative models have also been applied to time series anomaly detection, such as BeatGAN (Zhou et al., 2019), a variant of generative adversarial networks. In recent years, Anomaly Transformer (Xu et al., 2022) has introduced the correlation difference between normal points and abnormal points to improve the effect of anomaly detection. Dcdetector (Yang et al., 2023) uses patch learning of local information and permutation-invariant representation based on Anomaly Transformer to improve detection accuracy. D3R (Wang et al., 2023) addresses drift in time series by dynamically decomposing with data-time mix-attention and externally controlling the reconstruction bottleneck via noise diffusion. DMamba (Chen et al., 2024) proposes a selective state space model with a multi-stage detrending mechanism to enhance long-range dependency modeling and generalization in non-stationary Time Series Anomaly Detection. In addition, to capture the correlation between variables, graph structures are also used in time series anomaly detection. GSC_MAD (Zhang et al., 2024) uses graph structures for anomaly detection and achieves good results.

Memory networks were initially applied in natural language processing (Weston et al., 2014), leveraging reasoning components and long-term memory components to perform inference and learn how to utilize them jointly. The long-term memory is designed to support read and write operations, enabling it to be used for prediction tasks. In the context of question answering (QA), these models were explored where the long-term memory effectively serves as a (dynamic) knowledge base, with the output being a textual response. Later, an improved version of memory networks was proposed (Sukhbaatar et al., 2015), employing end-to-end training and utilizing a recurrent attention model to retrieve memory items. In recent years, memory networks have emerged as a powerful tool in various fields, particularly in computer vision. They are increasingly utilized for tasks such as video representation learning and text-to-image synthesis, demonstrating their versatility and effectiveness. Within the domain of anomaly detection, MemAE Gong et al. (2019) was pioneering in integrating memory networks into an autoencoder framework, specifically aimed at detecting anomalies in computer vision applications. Despite its innovative approach, MemAE's performance was limited by the lack of a dedicated mechanism for updating memory content.To overcome this shortcoming, MNAD Park et al. (2020) proposed a novel method for updating memory items by storing multiple patterns of normal behavior within the memory framework. While this method marked an improvement over MemAE, it still faced challenges; the memory was updated through a weighted sum of related queries, which made it difficult to effectively manage the incorporation of new information and ensure relevant learning.Building on these foundational ideas, MEMTO Song et al. (2024) was specifically designed for time series anomaly detection and introduced update gates to regulate the amount of new information that memory prototypes could absorb.

3 Proposed Method

First of all, the original time series $\mathcal{X}$ is divided into multiple subsequences by a sliding window, $\mathcal{X} = \{\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_a\}$. Each subsequence $\mathbf{X} \in \mathcal{X}$ is taken as one time series for training. Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_L\}$, where L is the length of the series and $\mathbf{x}_t \in \mathbb{R}^C$ is an observation vector at time t. MTSAD aims at learning knowledge from normal time series and generating labels $\mathbf{Y}_{test} = \{y_1, y_2, \ldots, y_{L_1}\}$ for unseen series $\mathbf{X}_{test} \in \mathbb{R}^{L_1 \times C}$, where $y_t \in \{0, 1\}$, 0 for normal and 1 for abnormal. For reconstruction-based model, anomaly scores are given by $s_t = \|\mathbf{x}_t - \hat{\mathbf{x}}_t\|_2$ where $\hat{\mathbf{x}}_t$ is the reconstructed observation of $\mathbf{x}_t$ with learning knowledge and $\|\cdot\|_2$ is the L2-norm. Finally, anomaly labels are determined by the anomaly threshold θ, $y_t = 1$ for $s_t \geq \theta$ and $y_t = 0$ otherwise.

This paper proposes H-PAD to learn normal knowledge from patches of different sizes in temporal domain (right part in Figure 2) and learn period prototypes in frequency domain (left part in Figure 2). $\mathbf{x}_t$ is reconstructed with both temporal and frequency knowledge. It not only prevents the occurrence of over-generalization, but also be able to discover different types of anomalies.

3.1 Learning Normal Patch Prototypes

To learn different scales of temporal information, H-PAD features different sizes of prototypes from different sizes of patches from normal times series. Different scales of local information are contained into different sizes of prototypes and different views of normal features are embedded into patch prototypes.

Average Pooling Given $\mathbf{X} \in \mathbb{R}^{L \times C}$, it is divided into several patches of size $z \in \{1, 2, \cdots, m\}$ without overlapping. All $\mathbf{x}_t$ in the same patch is averaged according to equation 1

$$\mathbf{x}_i^z = \frac{1}{z} \sum_{t=t_0}^{i \cdot z} \mathbf{x}_t \tag{1}$$

where $t_0 = (i-1) \cdot z + 1$. Thus a new series is generated to obtain $\mathbf{X}^z = \left\{\mathbf{x}_1^z, \mathbf{x}_2^z, \cdots, \mathbf{x}_{L_z}^z\right\}$, where $L_z = \left\lceil \frac{L}{z} \right\rceil$. Specifically, $\mathbf{X}^z$ for $z = 1$ is the original time series $\mathbf{X}$ which remains the original information after passing through the pooling layer with a pooling kernel size of 1. With average pooling, series from different patch sizes involve different width local information.

Update Patch Prototypes To learn prototypes in feature space, the generated series $\mathbf{X}^z$ are embedded into the feature space of higher dimension according to the Transformer Encoder

$$\mathbf{Q}^z = Encoder\left(\mathbf{X}^z\right) \tag{2}$$

where $\mathbf{Q}^z = \left\{\mathbf{q}_1^z, \mathbf{q}_2^z, \cdots, \mathbf{q}_{L_z}^z\right\} \in \mathbb{R}^{L_z \times D}\,(D > C)$, which is used to learn patch prototypes in different scales (Figure 3(a)). The detailed information about the Transformer Encoder can be found in Appendix B.

To obtain prototypes of normal patterns, an update gate ψ is used to update the prototypes. Since the normal patterns in the prototypes are derived from normal information, we reconstruct the initial prototype $\mathbf{b}_i^z$ using the similarity matrix v_{ij}^z between each prototype $\mathbf{b}_i^z$ and all normal features $\mathbf{q}^z$, as well as all the normal features $\mathbf{q}^z$. The reconstructed prototype $v_{ij}^z \mathbf{q}_j^z$ thus contains normal information. To update the prototypes, the update gate ψ is employed to determine how much of the original prototype information to retain and how much of the reconstructed prototype information to incorporate. The update gate ψ is constructed using two linear projections, $\mathbf{U}^z$ and $\mathbf{W}^z$, applied to the original prototype $\mathbf{b}_i^z$ and the reconstructed prototype $v_{ij}^z \mathbf{q}_j^z$, followed by a nonlinear activation function. Initializing randomly, patch prototypes $\mathbf{B}^z = \{\mathbf{b}_1^z, \mathbf{b}_2^z, \cdots, \mathbf{b}_M^z\} \in \mathbb{R}^{M \times D}$ are updated according to a update gate ψ (Song et al., 2024)

$$\mathbf{b}_i^z = (\mathbf{1}_D - \psi) \odot \mathbf{b}_i^z + \psi \odot \sum_{j=1}^{L_z} v_{ij}^z \mathbf{q}_j^z \tag{3}$$

where

$$\psi = \sigma\left(\mathbf{U}^z \mathbf{b}_i^z + \mathbf{W}^z \sum_{k=1}^{L_z} v_{ik}^z \mathbf{q}_k^z\right), \qquad v_{ij}^z = \frac{\exp\left(\left\langle \mathbf{b}_i^z, \mathbf{q}_j^z \right\rangle / \tau\right)}{\sum_{r=1}^{L_z} \exp\left(\left\langle \mathbf{b}_i, \mathbf{q}_r^z \right\rangle / \tau\right)} \tag{4}$$

and τ is the temperature parameter. $\mathbf{U}^z$ and $\mathbf{W}^z$ are linear transformation matrices, σ is the sigmoid activation function and $\odot$ takes element-by-element multiplication. With attention weights, patch prototypes are updated to remember normal features of the query series.

Query Reconstruction First of all, the updated patch prototypes $\mathbf{B}^z$ are taken to reconstruct the query series $\mathbf{Q}^z$. Embedding different widths of local normal information, the query series can be reconstructed to be $\hat{\mathbf{Q}}^z = \left\{\hat{\mathbf{q}}_1^z, \hat{\mathbf{q}}_2^z, \cdots, \hat{\mathbf{q}}_{L_z}^z\right\}$ according to

$$\hat{\mathbf{q}}_j^z = \sum_{k=1}^{M} w_{jk}^z \mathbf{b}_k, \quad \text{where} \quad w_{jk}^z = \frac{\exp\left(\left\langle \mathbf{q}_j^z, \mathbf{b}_k^z \right\rangle / \tau\right)}{\sum_{r=1}^{M} \exp\left(\left\langle \mathbf{q}_j^z, \mathbf{b}_r^z \right\rangle / \tau\right)} \tag{5}$$

w_{jk}^z is the attention weight of the query $\mathbf{q}_j^z$ for the patch prototype $\mathbf{b}_k^z$.

To reconstruct $\mathbf{Q} = \mathbf{Q}^{z_1}$ from different sizes of queries $\hat{\mathbf{Q}}^{z_1}, \hat{\mathbf{Q}}^{z_2}, \cdots, \hat{\mathbf{Q}}^{z_m}$, they should be integrated properly. Recalling the strategy of average pooling, local information of $\mathbf{q}_t$ is included into $\mathbf{q}_{t_j}^{z_j}$ where

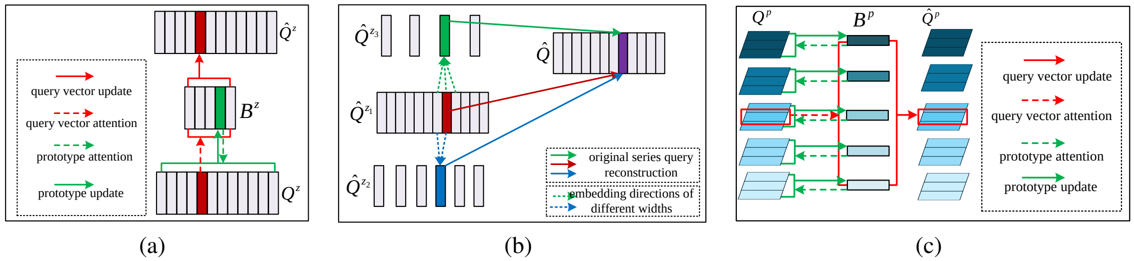


Figure 3: (a) Updating normal patch prototypes. (b) Query reconstruction with normal patch prototypes. (c) Learning period prototypes.

$(t_j - 1) \cdot z_j + 1 \leq t \leq t_j \cdot z_j \quad (j = 1, 2, \cdots, m)$. That is $\mathbf{q}_{t_j}^{z_j}$ contains the normal information of $\mathbf{q}_t$ in patch scale z_j. Naturally, $\hat{\mathbf{q}}_{t_j}^{z_j}$ is taken into account to reconstruct $\mathbf{q}_t$ (illustrated as in Figure 3(b)). In this paper, all related $\hat{\mathbf{q}}_{t_j}^{z_j}$ is integrated with a linear network to reconstruct $\mathbf{q}_t$:

$$\hat{\mathbf{q}}_t = Linear\left(ReLU\left(Linear\left(\hat{\mathbf{q}}_{t_1}^{z_1}, \cdots, \hat{\mathbf{q}}_{t_1}^{z_2}, \cdots, \hat{\mathbf{q}}_{t_1}^{z_m}\right)\right)\right). \tag{6}$$

Finally, $\hat{\mathbf{Q}} = (\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2, \cdots, \hat{\mathbf{q}}_L)$ is concatenated with $\mathbf{Q}^{z_1}$ and decoded into original space $\mathbb{R}^C$as the reconstructed series $\hat{\mathbf{X}}_z$ of $\mathbf{X}$ by normal patch prototypes.

3.2 Learning Period Prototypes

Most time series in the real world are multi-periodic, and these periods influence each other, presenting the overall variation tendency of the time series (Wu et al., 2023). In addition to normal patch prototypes, period prototypes of normal time series are learned to model different characteristics of different periodic patterns.

Period Division For periodical information of time series, Fast Fourier Transform (FFT) is carried out to obtain different periods(Wu et al., 2023). Given the original time series $\mathbf{X} \in \mathbb{R}^{L \times C}$, a set of periods $P = \{p_1, p_2, \cdots, p_k\}$ can be gained from

$$\mathbf{A} = Avg\left(Amp\left(FFT(\mathbf{X})\right)\right) \tag{7}$$

$$\{f_1, f_2, \ldots f_k\} = argTopk(\mathbf{A}), \quad p_i = \left\lceil \frac{L}{f_i} \right\rceil (i = 1, 2, \cdots, k) \tag{8}$$

where FFT and Amp represent Fast Fourier Transform and calculation of amplitude value respectively. $\mathbf{A}$ represents the calculated amplitudes and higher amplitudes contain more significant information. Therefore, we select the top-k amplitudes and derive the corresponding periods P based on them. With $p \in P$, the original series can be divided along timeline to get $N = \left\lceil \frac{L}{p} \right\rceil$ segments (zero-padding at the end). The similarity and periodicity of segments are taken into serious consideration along each variable. Thus, the divided time series is reshaped to $\mathcal{X}^p = \{\mathbf{X}_1^p, \mathbf{X}_2^p, \cdots, \mathbf{X}_C^p\}$ where $\mathbf{X}_j^p \in \mathbb{R}^{N \times p}$. To characterize each period, $\mathbf{X}_j^p$ is encoded to be $\mathbf{Q}_j^p \in \mathbb{R}^{N \times D}$ into features space by an encoder. Period prototypes will be learned for different periods and different variables from $\mathcal{Q}^p = \{\mathbf{Q}_1^p, \mathbf{Q}_2^p, \cdots, \mathbf{Q}_C^p\}$ in the feature space.

Update Period Prototypes For one observing variable of time series, one period prototype $\mathbf{b}^p \in \mathbb{R}^D$ is learned from one partition $\mathbf{Q}^p = \{\mathbf{q}_1^p, \mathbf{q}_2^p, \cdots, \mathbf{q}_N^p\}$ (shown as in Figure 3(c)). With randomly initialized

$\mathbf{b}^p$, the period prototype is updated with weighted segmented periods by

$$\mathbf{b}^p = (\mathbf{1}_D - \psi) \odot \mathbf{b}^p + \psi \odot \sum_{j=1}^{N} v_j^p \mathbf{q}_j^p \tag{9}$$

where

$$\psi = \sigma\left(\mathbf{U}^p \mathbf{b}^p + \mathbf{W}^p \sum_{k=1}^{N} v_k^p \mathbf{q}_k^p\right), \qquad v_j^p = \frac{\exp\left(\left\langle \mathbf{b}^p, \mathbf{q}_j^p \right\rangle / \tau\right)}{\sum_{r=1}^{N} \exp\left(\left\langle \mathbf{b}^p, \mathbf{q}_r^p \right\rangle / \tau\right)}. \tag{10}$$

Query Reconstruction After updating period prototypes, a set of period prototypes are gained, $\mathbf{B}^p = \{\mathbf{b}_1^p, \mathbf{b}_2^p, \cdots, \mathbf{b}_C^p\}$. Considering the correlation between variables, each query vector $\mathbf{q}_i^p$ can be recovered with the updated period prototypes $\mathbf{B}^p$ according to

$$\hat{\mathbf{q}}_i^p = \sum_{j=1}^{C} w_{ij}^p \mathbf{b}_j^p, \qquad \text{where} \quad w_{ij}^p = \frac{\exp\left(\left\langle \mathbf{q}_i^p, \mathbf{b}_j^p \right\rangle / \tau\right)}{\sum_{n=1}^{C} \exp\left(\left\langle \mathbf{q}_i^p, \mathbf{b}_n^p \right\rangle / \tau\right)} \tag{11}$$

The reconstructed period partition $\hat{\mathbf{Q}}^p = \{\hat{\mathbf{q}}_1^p, \hat{\mathbf{q}}_2^p, \cdots, \hat{\mathbf{q}}_N^p\}$ are concatenated with $\mathbf{Q}^p = \{\mathbf{q}_1^p, \mathbf{q}_2^p, \cdots, \mathbf{q}_N^p\}$ to produce $\tilde{\mathbf{Q}}^p = \{\tilde{\mathbf{q}}_1^p, \tilde{\mathbf{q}}_2^p, \cdots, \tilde{\mathbf{q}}_N^p\}$, $\tilde{\mathbf{q}}_i^p = (\mathbf{q}_i^p, \hat{\mathbf{q}}_i^p) \in \mathbb{R}^{2D}$. Collecting all reconstructed partitions of C variables, $\tilde{\mathcal{Q}}^p = \left\{\tilde{\mathbf{Q}}_1^p, \tilde{\mathbf{Q}}_2^p, \cdots, \tilde{\mathbf{Q}}_C^p\right\}$ is used to reconstruct $\mathbf{X}$ as $\hat{\mathbf{X}}_p$ with decoder.

3.3 Loss Function and Anomaly Scores

After reconstructing with patch and period prototypes respectively, $\mathbf{X}$ is reconstructed by the summation of $\hat{\mathbf{X}}_z$ and $\hat{\mathbf{X}}_p$, $\hat{\mathbf{X}} = \gamma \hat{\mathbf{X}}_z + (1-\gamma)\hat{\mathbf{X}}_p$. For learning prototypes stage, reconstruction loss (12) is minimized to learn normal prototypes,

$$L_{rec} = \|\mathbf{X} - \hat{\mathbf{X}}\|_F \tag{12}$$

where $\|\cdot\|_1$ is the Frobenius norm. Although normal patterns are learned from normal data, using too many prototypes for reconstruction may occasionally lead to well reconstructing anomalies. To prevent this, a sparsity constraint is applied, encouraging the model to use fewer prototypes for reconstruction, thereby reducing the likelihood of reconstructing anomalies by chance. Specifically, given w_{ji}^z as the weight matrix used for reconstruction with prototypes, minimizing the entropy loss ensures that a small number of weights approach 1 while the rest approach 0, effectively constraining the model to use fewer prototypes for reconstruction. For sparsity constraints on of patch prototypes, entropy loss of w_{ji}^z (Song et al., 2024) is minimized:

$$L_{ent} = \sum_{z=1}^{m} \sum_{j=1}^{L_z} \sum_{i=1}^{M} -w_{ji}^z \log\left(w_{ji}^z\right) \tag{13}$$

For period prototypes, they must well characterize the periodicity of the training time series. Therefore, a period loss in feature space is designed based on the distance between the period prototype $\mathbf{b}_i^p$ and the query vector $\mathbf{q}_{ij}^p$:

$$L_{prd} = \sum_{m=1}^{k} \sum_{i=1}^{C} \sum_{j=1}^{N} \|\mathbf{b}_i^{p_m} - \mathbf{q}_{ij}^{p_m}\|_2 \tag{14}$$

where $\mathbf{q}_{ij}^p$ is the j-th query of the i-th variable for the embedded p-th partition in feature space $\mathbb{R}^D$. In summary, the total loss function is a weighted combination of equation 12, equation 13, and equation 14:

$$L = \alpha_1 L_{rec} + \alpha_2 L_{ent} + \alpha_3 L_{prd} \tag{15}$$

where α_1, α_2, and α_3 are influence parameters of different loss parts.

To detecting anomalies, the protuberant deviations in both input space and feature space are designed into anomalies scores. In the input space, the reconstruction error is generally considered to be anomalies scores:

$$s_r(t) = \|\hat{\mathbf{x}}_t - \mathbf{x}_t\|_2 \tag{16}$$

In the feature space of patch prototypes, the distance between the query $\mathbf{q}_t$ and its most similar prototype $\mathbf{b}_{sim}^{z_j}$ is also considered into the anomaly score:

$$s_z(t) = \sum_{j=1}^{m} \frac{1}{z_j} \left\|\mathbf{q}_t - \mathbf{b}_{sim}^{z_j}\right\|_2 . \tag{17}$$

Remember that the larger the scale, the more different the prototype from the original query. An inverse proportional parameter $\frac{1}{z_j}$ is employed to adapt the influence of different scale of patch prototypes on $s_z(t)$. Analogously, in the feature space of period prototypes, the distance between the period query vector $\mathbf{q}_t^{p_j}$ and the most similar prototype $\mathbf{b}_{c_{sim}}^{p_j}$ is also designed into the anomaly score:

$$s_p(t) = \sum_{j=1}^{k} \left\|\mathbf{q}_t^{p_j} - \mathbf{b}_{c_{sim}}^{p_j}\right\|_1 \tag{18}$$

where $\|\cdot\|_1$ is L1-norm. Considering that both scores in feature space affects the reconstruction error, they are integrated into

$$s(t) = softmax(s_z(t) + \beta s_p(t)) \times s_r(t) \tag{19}$$

where β is the weight adapting parameters.

4 Experiments

4.1 Experimental Setup

Datasets Our model H-PAD is evaluated on five real-world multivariate datasets, namely, **MSL**, **SMAP**, **PSM**, **SMD**, and **SWaT**. In addition, two simulated datasets, **NIPS_TS_Water** and **NIPS_TS_Swan**, are used. The more detailed content of the dataset can be found in the appendix C.

Implementation details The training dataset is divided into 80% training set and 20% validation set. Common evaluation indicators are used for comprehensive comparison, such as accuracy (Acc), precision (Pre), recall (Rec), F1-score (F1), etc. We use the commonly used point adjustment technique for comparison (Shen et al., 2020).

4.2 Main results

We comprehensively compare our model with 16 baseline models. Table 1 gives the evaluation results of different baseline models and our model in five real datasets. It can be seen that our model H-PAD can achieve the best results in most datasets, with an F1 score of more than 95% on all datasets.

However, many works have demonstrated that PA can lead to faulty performance evaluations (Wang et al., 2023; Kim et al., 2021; Huet et al., 2022), and it is known that using PA can result in state-of-the-art performance even with random scores or random initialized non-trained models, making it impossible to conduct a fair comparison and assess the effectiveness of the models. To ensure a fair comparison between H-PAD and the baseline models, we used AUC-ROC and AUC-PR as evaluation metrics. As shown in Table 2, H-PAD achieves the best or second-best results on most datasets. Furthermore, H-PAD exhibited the highest average AUC-ROC score and the second-best AUC-PR score in all seven datasets, highlighting its effectiveness. Please refer to Appendix D for a more detailed description of the evaluation criteria.

Table 1: Comparison results on five real-world datasets with Pre (precision), Rec (recall), F1-score (%). The best results are marked in bold, and the second best results are marked in underline. ('Avg' means average, and 'A.T.' means Anomaly Transformer.)

Model	MSL			SMAP			PSM			SMD			SWaT			Avg
	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	F1
Isolation Forest	53.94	86.54	66.45	52.39	59.07	55.53	76.09	92.45	83.48	42.31	73.29	53.64	49.29	44.95	47.02	61.22
OC-SVM	59.78	86.87	70.82	53.85	59.07	56.34	62.75	80.89	70.67	44.34	76.72	56.19	45.39	49.22	47.23	60.25
LOF	47.72	85.25	61.18	58.93	56.33	57.60	57.89	90.49	70.61	56.34	39.86	46.68	72.15	65.43	68.62	60.94
DAGMM	89.60	63.93	74.62	86.45	56.73	68.51	93.49	70.03	80.08	67.30	49.89	57.30	89.92	57.84	70.40	70.18
MMPCACD	81.42	61.31	69.95	88.61	75.84	81.73	76.25	78.35	77.29	71.20	79.28	75.02	82.52	68.29	74.73	75.74
Deep-SVDD	91.92	76.63	83.58	89.93	56.02	69.04	95.41	86.49	90.73	78.54	79.67	79.10	80.42	84.45	82.39	80.97
THOC	88.45	90.97	89.69	92.06	89.34	90.68	88.14	90.99	89.54	79.76	90.95	84.99	83.94	86.36	85.13	88.01
ITAD	69.44	84.09	76.07	82.42	66.89	73.85	72.80	64.02	68.13	86.22	73.71	79.48	63.13	52.08	57.08	70.92
LSTM-VAE	85.49	79.94	82.62	92.20	67.75	78.10	73.62	89.92	80.96	75.76	90.08	82.30	76.00	89.50	82.20	81.24
OmniAnomaly	89.02	86.37	87.67	92.49	81.99	86.92	88.39	74.46	80.83	83.68	86.82	85.22	81.42	84.30	82.83	84.69
InterFusion	81.28	92.70	86.62	89.77	88.52	89.14	83.61	83.45	83.52	87.02	85.43	86.22	80.59	85.58	83.01	85.70
BeatGAN	89.75	85.42	87.53	92.38	55.85	69.61	90.30	93.84	92.04	72.90	84.09	78.10	64.01	87.46	73.92	80.24
A.T.	91.88	92.98	92.43	93.65	<u>99.47</u>	96.47	95.86	98.77	97.29	89.45	94.36	91.84	90.98	<u>95.56</u>	92.41	94.09
DCdetector	92.09	**98.89**	<u>95.37</u>	<u>94.42</u>	98.95	<u>96.63</u>	97.24	97.72	97.48	86.08	85.60	85.84	93.29	**100.00**	<u>96.53</u>	94.37
MEMTO	91.95	<u>97.23</u>	94.56	93.66	**99.73**	96.60	97.47	98.60	98.03	88.24	<u>96.16</u>	92.03	94.28	91.72	93.73	94.99
GSC_MAD	**94.19**	93.09	93.63	89.57	98.35	93.76	<u>97.97</u>	<u>99.14</u>	<u>98.89</u>	<u>92.25</u>	94.42	<u>93.32</u>	**96.73**	95.11	95.91	<u>95.10</u>
H-PAD	<u>94.05</u>	96.88	**95.45**	**96.00**	98.45	**97.21**	**98.82**	**99.41**	**99.12**	**92.86**	**98.20**	**95.45**	<u>94.34</u>	**100.00**	**97.09**	**96.86**

Table 2: Comparisons of AR (AUC-ROC) and AP (AUC-PR) on seven benchmark datasets.

Model	MSL		SMAP		PSM		SMD		SWaT		NIPS_TS_Water		NIPS_TS_Swan		Avg	
	AR	AP	AR	AP	AR	AP	AR	AP	AR	AP	AR	AP	AR	AP	AR	AP
DAGMM	56.47	11.72	52.30	**51.21**	49.85	30.74	63.38	10.06	81.01	<u>63.94</u>	77.24	9.67	46.47	29.29	60.96	29.51
THOC	58.09	14.60	40.37	10.32	<u>67.41</u>	<u>50.42</u>	66.18	13.80	**83.06**	**70.42**	73.16	<u>26.27</u>	77.27	<u>67.84</u>	66.50	**36.23**
LSTM-VAE	52.12	4.52	50.83	4.19	49.15	40.22	50.05	4.15	49.59	4.13	51.75	4.34	51.73	4.49	50.74	9.43
D3R	**65.26**	**16.99**	41.35	10.62	50.03	26.31	64.20	12.24	56.65	13.30	<u>80.32</u>	12.39	53.40	40.97	58.74	18.97
A.T.	48.72	10.64	49.67	12.50	48.56	29.42	47.28	3.70	29.40	8.82	33.46	1.48	43.49	28.62	42.94	13.59
DCdetector	50.06	10.61	48.87	12.48	49.83	27.64	48.77	**41.16**	49.74	11.60	50.53	1.72	48.50	31.71	49.47	19.56
MEMTO	49.99	10.48	**59.59**	<u>16.29</u>	49.75	26.96	<u>73.24</u>	10.35	45.41	11.45	60.96	4.21	51.12	49.06	55.72	18.40
DMamba	<u>61.54</u>	15.02	38.99	10.85	59.53	40.17	64.55	11.99	74.49	25.33	**96.93**	**46.32**	<u>77.84</u>	64.64	<u>67.69</u>	30.61
H-PAD	59.99	<u>15.06</u>	<u>59.13</u>	15.30	**75.01**	**51.83**	**76.49**	<u>14.05</u>	<u>81.54</u>	53.99	75.96	7.30	**81.66**	**74.31**	**72.83**	<u>33.12</u>

4.3 Ablation study

Effectiveness of module The different effectiveness of using patch prototypes and period prototypes is studies. As shown in Table 3, no matter which prototype is removed, the performance will decrease, and the performance decrease is the largest when the patch prototype is removed. Based on the comparison of these results, the effectiveness of using patch prototypes and period prototypes is demonstrated.

Abnormality Criteria Effectiveness of different abnormality score criteria is also studied. As shown in Table 4, when we only used the reconstruction error as the criterion, the F1 score dropped the most, with an average drop of 14.21%. Removing different evaluation criteria separately will result in different degrees of decline in results, which also proves the effectiveness of the abnormality criteria we used in this paper.

4.4 Discussion and analysis

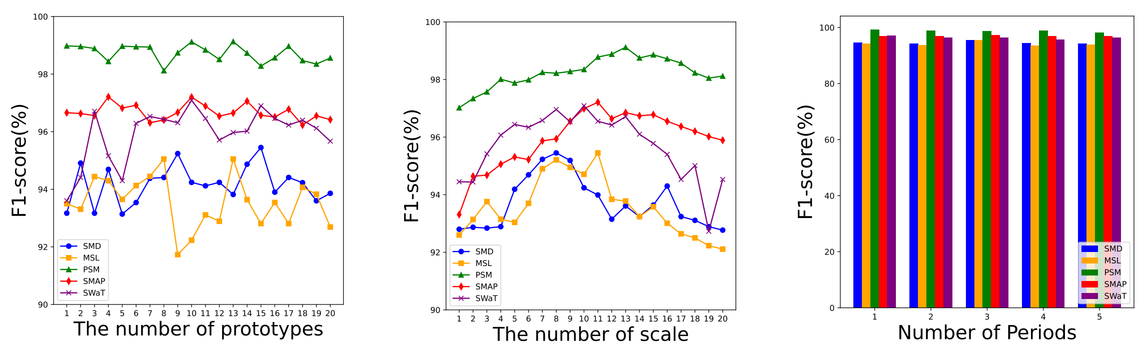
Parameter sensitivity analysis The impact of the numbers of prototypes, scales, and periods on H-PAD is analyzed numerically. It can be seen from Figure 4 that F1 has small variance for different number of prototypes per scale (Figure 4(a)) to exhibit the robustness of H-PAD, as well as the number of periods shown in Figure 4(c) . H-PAD can get optimal F1 for each dataset (Figure 4(b)) with the best scale number.

Table 3: Modules Effectiveness measured in F1-score (%). **Scale** is the patch prototype branch, and **Period** is the period prototype branch.

Scale	Period	MSL	SMAP	PSM	SWaT	SMD	Avg.
✗	✗	93.93	96.30	97.95	93.73	92.03	94.78
✓	✗	94.56	96.51	98.37	96.89	94.26	96.11
✗	✓	83.48	69.52	93.72	89.55	78.13	83.08
✓	✓	**95.45**	**97.21**	**99.12**	**97.09**	**95.45**	**96.86**

Table 4: Effectiveness of anomaly criterions. Comparison using F1 score (%).

s_{rec}	s_z	s_p	MSL	SMAP	PSM	SWaT	SMD	Avg.
✓	✗	✗	88.32	78.21	93.40	94.08	59.25	82.65
✓	✓	✗	93.18	96.73	98.24	96.78	93.24	95.63
✓	✗	✓	91.27	91.71	93.40	84.88	59.61	84.17
✗	✓	✓	93.18	96.47	97.85	96.62	89.84	94.79
✓	✓	✓	**95.45**	**97.21**	**99.12**	**97.09**	**95.45**	**96.86**



(a) (b) (c)

Figure 4: Sensitivity analysis of hyperparameters for H-PAD.

5 Conclusion

This paper proposes a new time series anomaly detection model H-PAD. Different prototypes-patch prototype and period prototype is utilized to detect different long-term and short-term anomalies. Furthermore, we use data of different scales to capture short-term changes, and use data of different periods to capture periodic information, so that time series can be modeled using long-term and short-term normal prototypes. Due to the incorporation of multi-scale information and multi-period information, our model achieves superior results but but its training time, number of model parameters, and GPU memory required for training are higher compared to other models. In future work, we plan to optimize the overall framework to improve efficiency, reducing training time and memory consumption without compromising performance. Additionally, we aim to conduct experiments on datasets from a broader range of domains to further validate the robustness of H-PAD.

REFERENCES

Ahmed Abdulaal, Zhuanghua Liu, and Tomer Lancewicki. Practical approach to asynchronous multivariate time series anomaly detection and localization. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery Data mining*, pp. 2485–2494, 2021.

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93–104, 2000.

Junqi Chen, Xu Tan, Sylwan Rahardja, Jiawei Yang, and Susanto Rahardja. Joint selective state space model and detrending for robust time series anomaly detection. *IEEE Signal Processing Letters*, 31:2050–2054, 2024.

Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. Time-series representation learning via temporal and contextual contrasting. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 2352–2359, 2021.

Dong Gong, Lingqiao Liu, Vuong Le, Budhaditya Saha, Moussa Reda Mansour, Svetha Venkatesh, and Anton Van Den Hengel. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1705–1714, 2019.

Alexis Huet, Jose Manuel Navarro, and Dario Rossi. Local evaluation of time series anomaly detection algorithms. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 635–645, 2022.

Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining & data mining*, pp. 387–395, 2018.

Neha Kant and Manish Mahajan. Time-series outlier detection using enhanced k-means in combination with pso algorithm. In *Engineering Vibration, Communication and Information Processing: ICoEVCI 2018, India*, pp. 363–373. Springer, 2019.

Siwon Kim, Kukjin Choi, Hyun-Soo Choi, Byunghan Lee, and Sungroh Yoon. Towards a rigorous evaluation of time-series anomaly detection. In *AAAI Conference on Artificial Intelligence*, 2021.

Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks. In *International conference on artificial neural networks*, pp. 703–716. Springer, 2019.

Zhihan Li, Youjian Zhao, Jiaqi Han, Ya Su, Rui Jiao, Xidao Wen, and Dan Pei. Multivariate time series anomaly detection and interpretation using hierarchical inter-metric and temporal embedding. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 3220–3230, 2021.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth IEEE international conference on data mining*, pp. 413–422. IEEE, 2008.

Daehyung Park, Yuuna Hoshi, and Charles C Kemp. A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. *IEEE Robotics and Automation Letters*, 3(3):1544–1551, 2018.

Hyunjong Park, Jongyoun Noh, and Bumsub Ham. Learning memory-guided normality for anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14372–14381, 2020.

Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

Lifeng Shen, Zhuocong Li, and James Kwok. Timeseries anomaly detection using temporal hierarchical one-class network. *Advances in Neural Information Processing Systems*, 33:13016–13026, 2020.

Youjin Shin, Sangyup Lee, Shahroz Tariq, Myeong Shin Lee, Okchul Jung, Daewon Chung, and Simon S Woo. Itad: integrative tensor-based anomaly detection system for reducing false positives of satellite systems. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pp. 2733–2740, 2020.

Junho Song, Keonwoo Kim, Jeonglyul Oh, and Sungzoon Cho. Memto: Memory-guided transformer for multivariate time series anomaly detection. *Advances in Neural Information Processing Systems*, 36, 2024.

Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2828–2837, 2019.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. *Advances in neural information processing systems*, 28, 2015.

Jian Tang, Zhixiang Chen, Ada Wai-Chee Fu, and David W Cheung. Enhancing effectiveness of outlier detections for low density patterns. In *Advances in Knowledge Discovery and Data Mining: 6th Pacific-Asia Conference, PAKDD 2002 Taipei, Taiwan, May 6–8, 2002 Proceedings 6*, pp. 535–548. Springer, 2002.

David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54:45–66, 2004.

Chengsen Wang, Zirui Zhuang, Qi Qi, Jingyu Wang, Xingyu Wang, Haifeng Sun, and Jianxin Liao. Drift doesn't matter: Dynamic decomposition with diffusion reconstruction for unstable multivariate time series anomaly detection. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations*, 2023.

Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Anomaly transformer: Time series anomaly detection with association discrepancy. In *International Conference on Learning Representations*, 2022.

Takehisa Yairi, Naoya Takeishi, Tetsuo Oda, Yuta Nakajima, Naoki Nishimura, and Noboru Takata. A data-driven health monitoring method for satellite housekeeping data based on probabilistic clustering and dimensionality reduction. *IEEE Transactions on Aerospace and Electronic Systems*, 53(3):1384–1401, 2017.

Yiyuan Yang, Chaoli Zhang, Tian Zhou, Qingsong Wen, and Liang Sun. 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*, pp. 3033–3045, 2023.

Zhen Zhang, Zhiqiang Geng, and Yongming Han. Graph structure change-based anomaly detection in multivariate time series of industrial processes. *IEEE Transactions on Industrial Informatics*, 2024.

Bin Zhou, Shenghua Liu, Bryan Hooi, Xueqi Cheng, and Jing Ye. Beatgan: Anomalous rhythm detection using adversarially generated time series. In *IJCAI*, volume 2019, pp. 4433–4439, 2019.

Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International conference on learning representations*, 2018.

A BACKGROUND

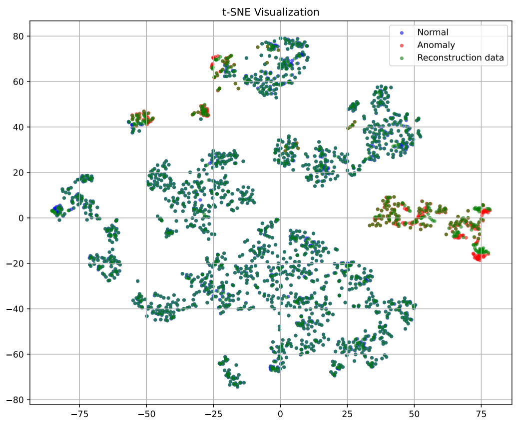
Time series anomaly detection is an unsupervised task, where normal data is used for reconstruction during the training phase. Since the model is trained on normal data, it learns to reconstruct the time series using normal features. In the testing phase, anomalous data is reconstructed using the normal features learned by the model, which transforms the anomalies into normal patterns. As a result, large reconstruction errors are observed at the anomalous points, allowing anomalies to be identified. However, if the model's reconstruction ability is too strong, it may reconstruct anomalous data as normal, making it difficult to detect anomalies. This is known as the overgeneralization problem.

By representing the test data with a t-SNE plot, as shown in Figure5(a) and Figure5(c), it is evident that the reconstructed data points are very close to the anomalous points of the original data. Due to the overgeneralization problem, it becomes challenging to identify anomalies. To address this, the model learns normal patterns from normal data as prototypes during the training phase. In the testing phase, these learned normal prototypes are used to reconstruct the test data. Since the prototypes only contain normal features, the reconstructed data will exhibit normal characteristics. Finally, to leverage both the normal features and the original features of the test data, the reconstructed normal features are concatenated with the original features and fed into a decoder. The reconstructed normal features suppress the anomalous features, resulting in the final normal reconstructed data.

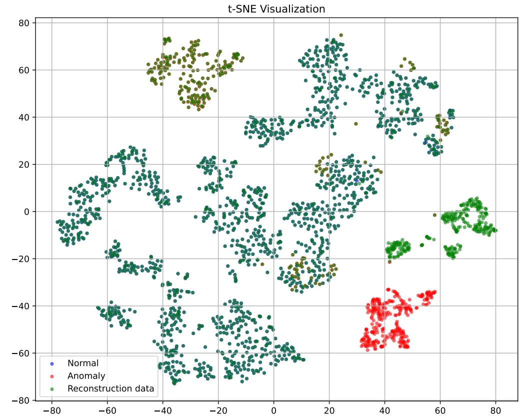
H-PAD leverages prototypes of different patches and different periods. This approach not only suppresses point anomalies but also handles short-term and periodic anomalies. For point anomalies, the differences between the anomalies and the normal data are evident, and using only normal point prototypes can effectively suppress point anomalies, enabling anomaly detection. For short-term anomalies, which may manifest as brief data fluctuations or sudden changes over a short period, single-point prototypes often fail to capture such short-term variations as they rely on trends and changes across multiple data points. By learning normal prototypes of varying patch sizes, both the local normal information and the trend information of normal patterns can be utilized, enabling the normal reconstruction of short-term anomalies. The same applies to periodic anomalies; single-point normal prototypes typically cannot capture periodic anomalies due to a lack of consideration for the periodic changes in the time series. Thus, normal prototypes for different periods are required for reconstruction. As shown in Figure5(b) and Figure5(d), after reconstruction using different normal prototypes in H-PAD, the reconstructed data is closer to the normal data and farther from the anomalous data. This effectively distinguishes anomalies, allowing the detection of whether the data is anomalous.

B ENCODER

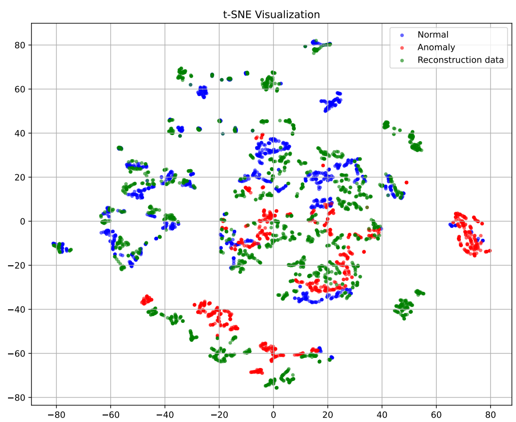
To make the data operation clearer, this section explains the working principle of the encoder part.



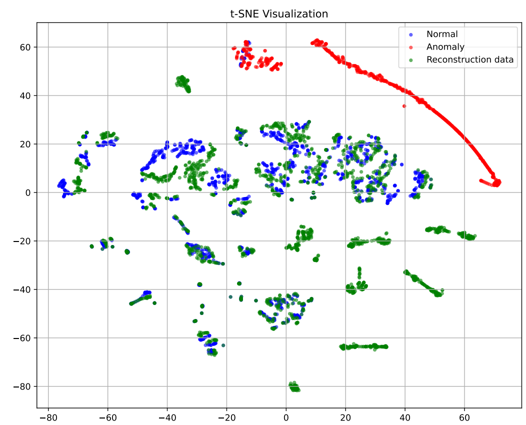
(a) Anomaly Transformer reconstructs PSM series



(b) H-PAD reconstructs PSM series



(c) Anomaly Transformer reconstructs SWAT series



(d) H-PAD reconstructs SWAT series

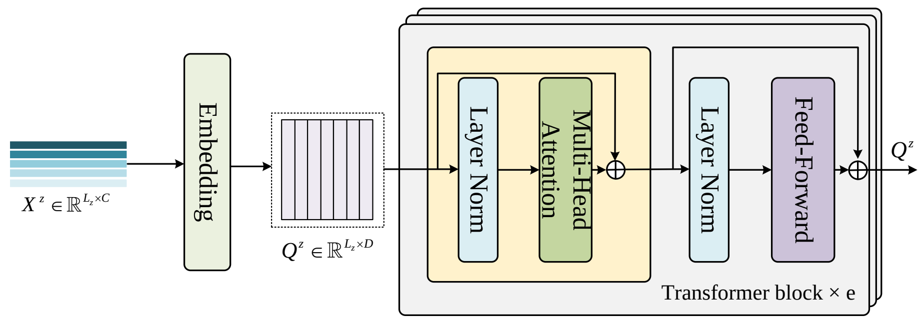
Figure 5: Overgeneralization visualization. Figures a and c are t-SNE graphs of the original data and the data reconstructed by Anomaly Transformer, and Figures b and d are t-SNE graphs of the original data and the data reconstructed by H-PAD. Blue points are normal data points, red points are abnormal data points, and green points are reconstructed data points.

To learn prototypes in feature space, different scale series are embedded into a higher feature space with Transformer Encoder(Figure 6(a)):

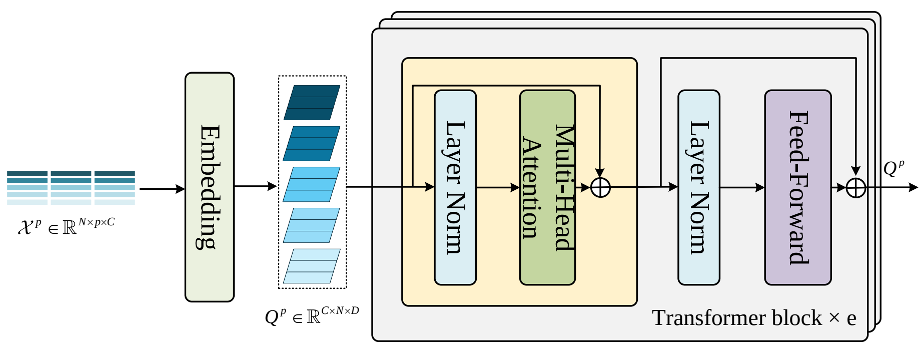
$$\mathbf{Q}^z = Transformer\left(Embedding\left(\mathbf{X}^z\right)\right) \tag{20}$$

where $\mathbf{Q}^z = \left\{\mathbf{q}_1^z, \mathbf{q}_2^z, \cdots, \mathbf{q}_{L_z}^z\right\} \in \mathbb{R}^{L_z \times D}\,(D > C)$. Specifically, the variable dimension C of the time series at each scale is embedded into a high-dimensional space D through a linear layer, resulting in the query vector $\mathbf{Q}^z$ for each scale. Then, $\mathbf{Q}^z$ is fed into e layers of Transformer blocks to capture long-term temporal dependencies, producing the final query vector $\mathbf{Q}^z$, which is used to learn the prototypes of different patches.

To characterize each period, $\mathbf{X}_j^p \in \mathbb{R}^{N \times p}$ is encoded to be $\mathbf{Q}_j^p \in \mathbb{R}^{N \times D}$ into features space by an encoder(Figure 6(b)), $\mathbf{X}_j^p$ represents the j-th variable data of $\mathbf{X}^p$. N is the number of periods and p is the size of the period. Unlike the previous approach where time series of different scales were embedded into a high-dimensional space D along the variable dimension, here, for different periods p of the time series, each variable contains distinct periodic information. Therefore, each period is embedded separately into a high-dimensional space D. A Transformer is then used to learn the correlations between periods across variables. Period prototypes will be learned for different periods and different variables from $\mathcal{Q}^p = \{\mathbf{Q}_1^p, \mathbf{Q}_2^p, \cdots, \mathbf{Q}_C^p\}$ in the feature space.



(a) learn long-term dependencies at different scales



(b) learn correlations between periods

Figure 6: Transformer Encoder.

C DATASETS

Our model H-PAD is evaluated on five real-world multivariate datasets. The specific description of the dataset is shown in Table 5.

MSL(Mars Science Laboratory rober) Hundman et al. (2018). MSL is collected by NASA shows the status of sensor data and actuator data of the Mars rover, and the variable dimension of this dataset is 55 dimensions.

SMAP(Soil Moisture Active Passive Satellite) Hundman et al. (2018). SMAP is also a dataset released by NASA, and its dimension is 38 dimensions.

PSM(Pooled Server Metrics) Abdulaal et al. (2021). PSM is a public dataset from eBay Server Machines with 25 dimensions.

Table 5: The dataset used in this study. Train and Test represent the number of time points in the training dataset and the test dataset, respectively. AR represents the anomaly rate.

Dataset	Dims	Train	Test	AR (%)
SWaT	51	475200	449919	12.14
PSM	25	132418	87841	27.76
MSL	55	58317	73729	10.48
SMAP	25	135183	427617	12.83
SMD	38	708405	708420	4.16
NIPS_TS_Water	9	69260	69261	1.1
NIPS_TS_Swan	38	60000	60000	32.60

SMD(Server Machine Dataset) Su et al. (2019). SMD is server data collected by a large Internet company in 5 weeks, with 38 dimensions.

SWaT(Secure Water Treatment) Li et al. (2019). SWaT is a dataset obtained by running six phases of infrastructure continuously for 11 days, with 51 dimensions.

NIPS_TS_Water. NIPS_TS_Water is a data set for water quality testing of drinking water, with 9 dimensions.

NIPS_TS_Swan. NIPS_TS_Swan is a multivariate time series extracted from vector magnetograms of the solar photosphere, with 38 dimensions.

D EVALUATION CRITERIA

This section introduces the evaluation metrics used to assess the performance of the model in this study. It is common to use traditional point-based information retrieval measures, such as Precision, Recall, and F1-score, to assess the quality of methods by thresholding the anomaly score to mark each point as an anomaly or not.

True Positive (TP): The number of points correctly identified as anomalies.

True Negative (TN): The number of points correctly identified as normal.

False Positive (FP): The number of points incorrectly identified as anomalies.

False Negative (FN): The number of points incorrectly identified as normal.

Precision: The proportion of data points predicted to be abnormal that are actually abnormal is calculated as follows:

$$precision = \frac{TP}{TP + FP}. \tag{21}$$

Recall: The proportion of data points that are correctly predicted to be abnormal among the data points that are actually abnormal is:

$$recall = \frac{TP}{TP + FN}. \tag{22}$$

F1-score: In time series anomaly detection, the F1 score is an indicator used to comprehensively evaluate the performance of the model. It is the harmonic mean of precision and recall. The formula for the F1 score is as follows:

$$F1 - score = 2 \cdot \frac{precision \cdot recall}{precision + recall}. \tag{23}$$

However, mapping discrete labels into continuous data introduces inherent limitations, particularly when evaluating range-based anomalies. While these classical metrics are effective for tasks that assess each sample independently, they fall short for time series datasets, where the temporal dimension is intrinsically continuous. Another notable limitation is the need to define a threshold on the anomaly scores generated by the detection method to classify each time series point as normal or anomalous. However, selecting an appropriate threshold is often challenging and prone to inaccuracies, making it a non-trivial task.

In time series anomaly detection, AUC-ROC and AUC-PR are commonly used metrics to evaluate model performance. To ensure that the evaluation is not influenced by the choice of thresholds, these metrics are employed to measure the model's performance across various threshold settings.

AUC-ROC: AUC-ROC (Area Under the Receiver Operating Characteristic Curve) is the area under the ROC curve. The ROC curve is a curve drawn with the false positive rate (FPR) as the horizontal axis and the true positive rate (TPR) as the vertical axis. It measures the model's ability to distinguish between positive and negative samples, and is particularly suitable for data with a balanced class distribution.

$$TPR = \frac{TP}{TP + FN}. \tag{24}$$

$$FPR = \frac{FP}{FP + TN}. \tag{25}$$

AUC-PR: AUC-PR (Area Under the Precision-Recall Curve) is the area under the Precision-Recall curve. The Precision-Recall curve is a curve drawn with the recall rate (Recall) as the horizontal axis and the precision rate (Precision) as the vertical axis. It is more suitable for datasets with imbalanced categories because abnormal samples in anomaly detection often account for a small proportion.

Afterwards,we use the affiliation metrics, an extension of the classical precision/recall for time series anomaly detection that is local, parameter-free, and applicable generically on both point and range-based anomalies. The metrics leverage measures of duration between ground truth and predictions, and have thus an intuitive interpretation.

To evaluate the overall performance of H-PAD compared to another memory-based model MEMTO, we use Aff-P and Aff-R, which calculate the average directed distance between sets to assess model performance. On one hand, if most anomalies in the predicted set significantly overlap with the ground truth anomalies, the average directed distance from the predicted set to the ground truth set will be smaller, resulting in a higher Aff-P for the model. On the other hand, if most anomalies in the ground truth set are covered by the predicted set, the average directed distance from the ground truth set to the predicted set will also be shorter, leading to a higher Aff-R for the model.

E Training efficiency analysis

H-PAD uses the ADAM optimizer with an initial learning rate of 10^{-4}. The training process is stopped early within 8 epochs with a batch size of 32. All experiments are implemented in Pytorch using a single

NVIDIA GeForce RTX 4090 24GB GPU. The efficiency of the H-PAD training model is compared with another memory model MEDMTO. The results are shown in Table 6. Since H-PAD learns patch prototypes of time series of different scales and period prototypes of different periods, its efficiency is much higher than MEMTO. MACs is the total number of multiplication-accumulation operations, which is used to measure the computational complexity of the neural network model. Epoch Time is the training time per epoch, in seconds. Max Memory Allocated is the maximum GPU memory usage during training. Total Parameters is the total number of parameters in the neural network model, including weights, biases, and other parameters.

Table 6: Training efficiency analysis.

Dataset	Method	MACs	NPARAMS	EROCH TIME	MAX MEMORY(GB)
MSL	MEMTO	10415226880	5955182	1.46	1.04
	H-PAD	90085779336	36345527	10.91	10.98
SMAP	MEMTO	10261626880	5862962	1.39	1.78
	H-PAD	35983470152	23556227	19.58	9.36
PSM	MEMTO	10261626880	5862962	1.05	2.58
	H-PAD	32452186440	22462860	10.70	3.35
SWaT	MEMTO	10394746880	5942886	3.65	4.37
	H-PAD	72704662360	35411680	54.86	4.72
SMD	MEMTO	10328186880	5902924	11.17	1.47
	H-PAD	41111111808	20404825	81.89	6.33

F More Experiments

To further demonstrate the effectiveness of using patch prototypes and using period prototypes, we compared multiple indicators with the memory model MEMTO (Song et al., 2024) that only learns point normal prototypes. The results are shown in Table 7. Affiliation precision(Aff-P) and recall(Aff-R) are calculated based on the distance between ground truth and prediction events. VUS metric takes anomaly events into consideration based on the receiver operator characteristic(ROC) curve. R_A_R and R_A_P are Range-AUC-ROC and Range-AUC-PR, respectively, representing the two scores obtained under the ROC curve and PR curve according to the label transformation. V_ROC and V_PR are the volumes under the ROC curve and PR curve, respectively.

Table 7: Comparion results with MEMTO with different metrics on real-world datasets.

Dataset	Method	Acc	F1	Aff-P	Aff-R	R_A_R	R_A_P	V_ROC	V_PR
MSL	MEMTO	98.09	93.54	51.43	96.00	90.56	88.35	88.71	86.73
	H-PAD	**99.03**	**95.45**	**55.98**	**96.25**	**91.34**	**88.66**	**89.94**	**87.91**
SMAP	MEMTO	99.07	96.60	**52.89**	96.92	94.45	93.48	93.20	92.40
	H-PAD	**99.28**	**97.21**	52.46	**98.87**	**96.83**	**94.13**	**95.86**	**93.30**
PSM	MEMTO	98.79	98.03	56.86	74.00	89.69	**94.20**	88.71	**92.57**
	H-PAD	**99.51**	**99.12**	**64.29**	**84.79**	**92.91**	92.42	**90.65**	91.70
SWaT	MEMTO	98.44	93.73	59.04	93.41	92.01	89.16	92.09	89.23
	H-PAD	**99.22**	**97.09**	**60.40**	**97.54**	**98.28**	**95.88**	**98.31**	**95.91**
SMD	MEMTO	99.18	92.03	56.19	86.93	74.69	71.21	74.95	71.48
	H-PAD	**99.60**	**95.45**	**68.91**	**93.22**	**81.02**	**78.86**	**82.02**	**79.85**

In addition, to further explore the impact of different hyperparameters on model performance, we conducted more parameter sensitivity experiments. The results are shown in the table 8, table 9, table 10, table 11, table 12, figure 8(a), figure 8(b) and figure 7.

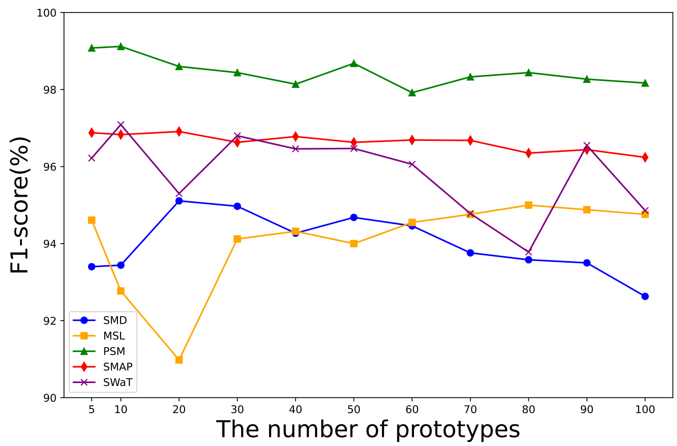


Figure 7: Sensitivity experiments with a larger number of prototypes.

Table 8: The impact of different sizes of hyperparameters γ on the model. The result is F1 score (%).

γ	MSL	SMAP	PSM	SWaT	SMD
0.1	94.06	97.01	98.99	96.33	89.37
0.3	93.14	96.93	98.95	94.43	93.80
0.5	95.45	97.21	99.12	97.09	95.45
0.7	93.20	96.30	98.97	96.68	94.16
0.9	92.69	96.28	99.04	95.97	95.15

G VISUALIZATION ANALYSIS

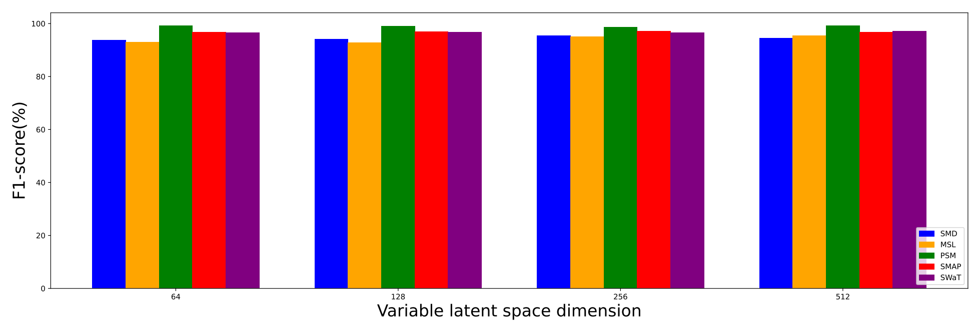
Anomaly detection results is visualized in Figure9 to to verify the effectiveness of the proposed H-PAD. It can be seen that H-PAD can assign higher anomalies scores to anomalies correctly and better detect various anomalies, which further illustrates the effectiveness of our model.

Table 9: The impact of different sizes of hyperparameters β on the model. The result is F1 score (%).

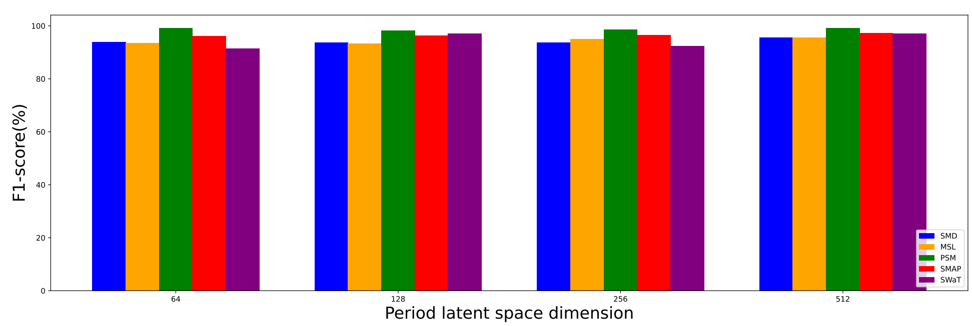
β	MSL	SMAP	PSM	SWaT	SMD
1	93.07	95.33	98.23	95.01	91.33
0.1	93.99	96.72	98.77	96.39	93.47
0.01	95.45	97.21	99.12	97.09	95.45
0.001	94.93	97.01	99.05	96.87	94.69

Table 10: The impact of different sizes of hyperparameters α_1 on the model. The result is F1 score (%).

α_1	MSL	SMAP	PSM	SWaT	SMD
0	94.92	96.83	98.13	96.46	90.95
0.1	93.32	96.59	98.13	94.66	95.00
0.5	94.00	96.63	98.68	94.66	93.59
0.8	93.53	96.42	98.95	96.48	94.61
1	95.45	97.21	99.12	97.09	95.45



(a) Variable latent space dimension D.



(b) Period latent space dimension D.

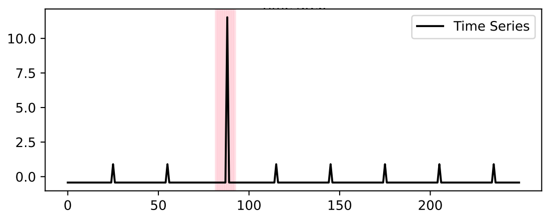
Figure 8: Hyperparameter sensitivity analysis of the latent space dimension.

Table 11: The impact of different sizes of hyperparameters α_2 on the model. The result is F1 score (%).

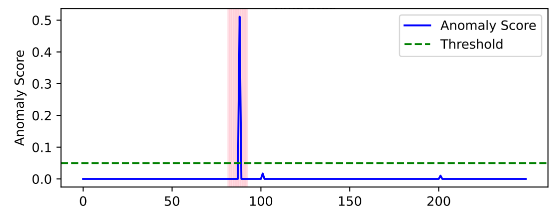
α_2	MSL	SMAP	PSM	SWaT	SMD
0	93.06	96.62	99.00	95.95	93.12
0.1	94.10	96.35	98.63	96.57	93.70
0.01	93.12	96.75	98.88	96.57	94.48
0.05	94.10	96.42	98.61	95.80	94.57
0.005	95.45	97.21	99.12	97.09	95.45
0.001	93.10	96.82	98.92	95.07	94.99

Table 12: The impact of different sizes of hyperparameters α_3 on the model. The result is F1 score (%).

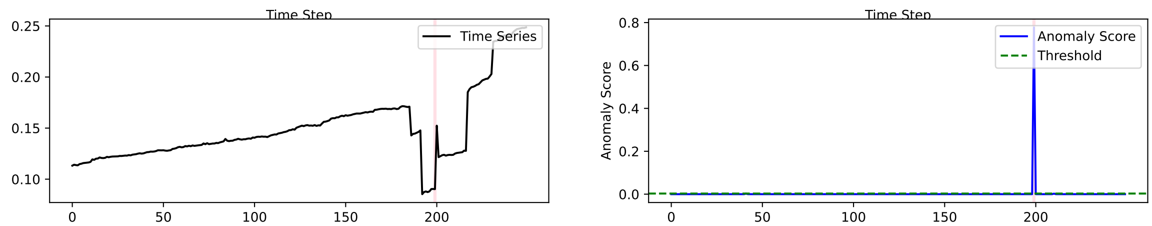
α_3	MSL	SMAP	PSM	SWaT	SMD
0	93.41	97.11	98.94	94.17	94.44
0.1	92.72	96.96	98.95	92.65	94.64
0.01	93.02	96.73	98.60	94.80	94.55
0.001	93.19	96.57	99.09	95.94	94.34
0.0001	95.45	97.21	99.12	97.09	95.45



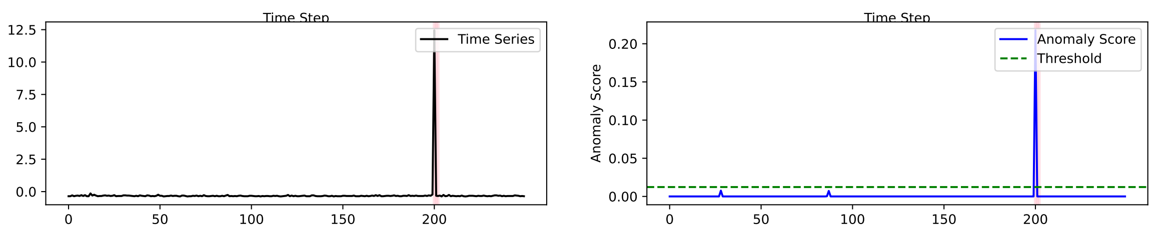
(a) Period anomaly



(b) Anomaly score

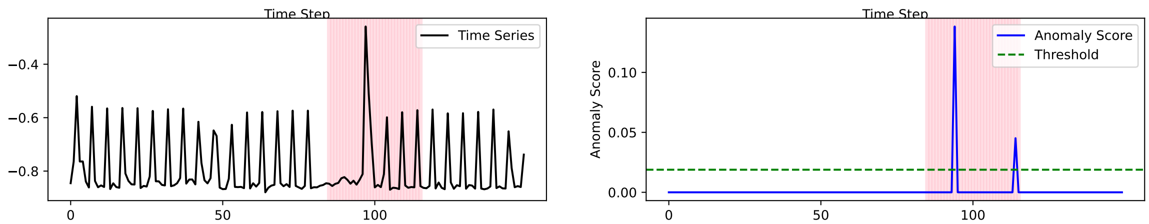


(c) Contextual anomaly

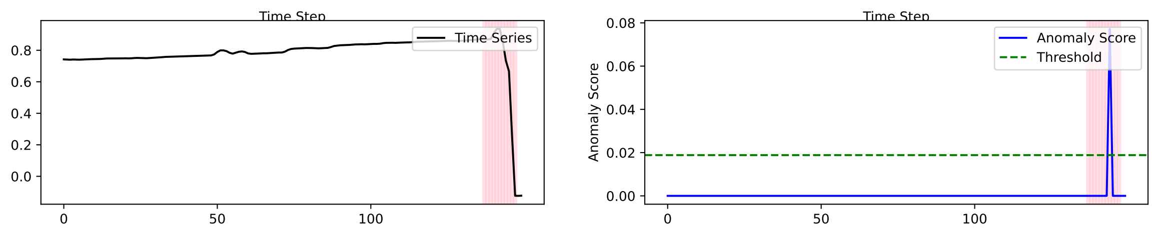


(d) Point anomaly

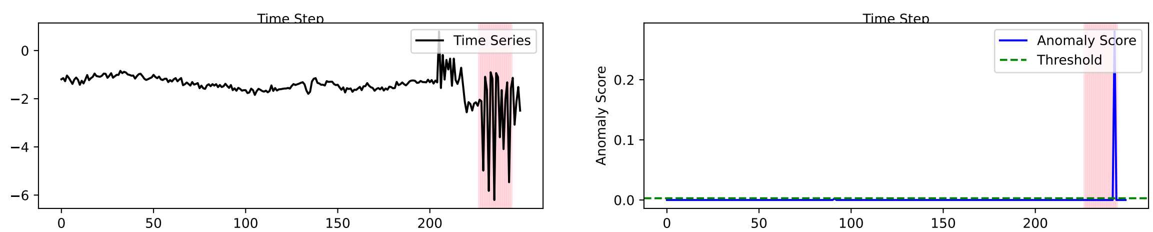
Figure 9: Anomaly visualization (Part 1). For each anomaly, such as periodic anomaly, the left side is the anomaly instance and the right side is the corresponding anomaly score.



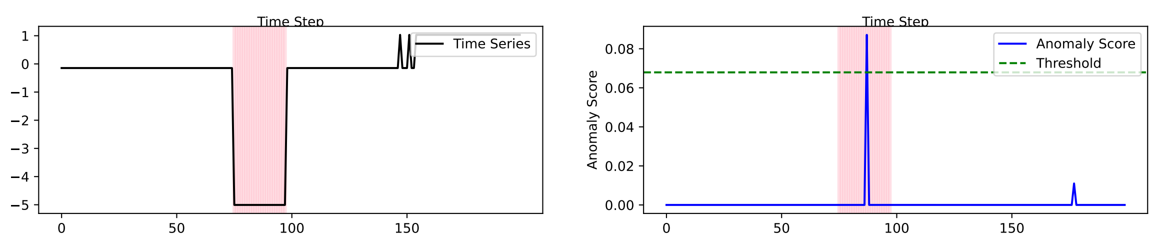
(a) Shapelet anomaly



(b) Tread anomaly



(c) Collective anomaly



(d) Global subsequence anomaly

Figure 9: Anomaly visualization (Part 2). For each anomaly, such as periodic anomaly, the left side is the anomaly instance and the right side is the corresponding anomaly score.