Engineering Applications of Artificial Intelligence 158 (2025) 111308

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Research paper

MemMambaAD: Memory-augmented state space model for multivariate time series anomaly detection

Gang Li [a,b], Mingchao Ge [a,b], Jin Wan [a,b], Delong Han [a,b], Min Li [a,b], Mingle Zhou [a,b,c,*]

[a] Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan, China
[b] Shandong Provincial Key Laboratory of Computing Power Internet and Service Computing, Shandong Fundamental Research Center for Computer Science, Jinan, China
[c] SHANDONG SCICOM Information and Economy Research Institute Co., Ltd., Jinan, China

ARTICLE INFO

Keywords:
Time series anomaly detection
State space model
Dynamic memory update mechanism

ABSTRACT

Multivariate time series anomaly detection focuses on recognizing abnormal patterns to reduce system failures and improve production efficiency and product quality. Accurately detecting anomalies in data remains challenging because existing reconstruction-based methods are prone to overfitting. Recently, reconstruction methods guided by memory modules have been used to address this issue. However, these methods still suffer from insufficient feature extraction of time series prototypes and inadequate storage of normal sample prototype patterns in memory modules. To address these issues, we propose a memory-augmented state space model for multivariate time series anomaly detection. Specifically, we introduce a sequence decomposition state space model-temporal convolutional encoder, which independently extracts trend and seasonal features of multivariate time series in global and local manner, capturing the intrinsic patterns of time series more comprehensively. In addition, we propose a dynamic memory update mechanism, which flexibly updates the memory item through the memory selection mechanism, to more accurately record the prototype pattern of normal samples to improve anomaly detection performance. Extensive experiments on five benchmark datasets demonstrate that our method achieves state-of-the-art performance and reduces memory usage compared with other methods.

1. Introduction

Multivariate time series anomaly detection aims to determine whether an anomaly occurs at each timestamp in a multivariate time series. With the widespread application of sensors in the real world, continuously operating systems generate multiple continuous measurements, such as those from industrial and network equipment (Abdulaal et al., 2021). These continuous measurements record the operation of the system. By performing anomaly detection on these time series, system failures can be discovered in a timely manner, which is significant for ensuring operational safety and reducing economic losses (Yang et al., 2021). Multivariate time series data is highly complex, including time dependence, high dimensionality, and non-stationarity. Anomalies in time series are typically varied, including point anomalies and collective anomalies (Li and Jung, 2023). The complexity of multivariate time series data and the diversity of anomalies make the time series anomaly detection task challenging and often lead to suboptimal performance in traditional methods. Additionally, due to the difficulty in labeling abnormal data in multivariate time series, unsupervised methods (Zong et al., 2018; Ruff et al., 2018; Park et al., 2018) are commonly utilized in anomaly detection.

Unsupervised time series anomaly detection methods typically involve comparing the calculated anomaly score of each data point with a threshold to determine anomalies. Current unsupervised time series anomaly detection methods can be divided into three categories: distance-based methods (Schölkopf et al., 1999), prediction-based methods (Shen et al., 2020; Deng and Hooi, 2021), and reconstruction-based methods (Li et al., 2021; Xu et al., 2021; Tuli et al., 2022). Distance-based methods rely on the similarity between time series data points and use distance metrics as criteria to determine anomalies. However, these methods do not perform well in high-dimensional time series data. Prediction-based methods usually involve building a time series forecasting model to estimate future values and obtaining anomaly scores based on the difference between the predicted and actual values. The accurate prediction of future data

* Correspondence to: Faculty of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Sciences), Jinan, 250353, China.
E-mail addresses: lig@qlu.edu.cn (G. Li), 10431230064@stu.qlu.edu.cn (M. Ge), wanj@qlu.edu.cn (J. Wan), handl@qlu.edu.cn (D. Han), limin@qlu.edu.cn (M. Li), zhouml@qlu.edu.cn (M. Zhou).

https://doi.org/10.1016/j.engappai.2025.111308
Received 12 October 2024; Received in revised form 5 April 2025; Accepted 25 May 2025
Available online 13 June 2025

points may be challenging when the data exhibits nonlinear or complex patterns. Reconstruction-based methods are the current mainstream approach and obtain anomaly scores by comparing reconstructed data with original data. Although these methods have made encouraging progress, they are susceptible to overfitting, leading to model misclassification (Zhang et al., 2022). For example, in the anomaly detection process in the hydrogen energy field, the reconstruction capability of the model may generalize to abnormal data, resulting in missed detections and losses.

MemAugUTransAD (Qin et al., 2023) employs multiple memory mechanisms to mitigate the generalization of the method to abnormal data. Similarly, MEMTO (Song et al., 2024) alleviates overfitting by employing a Transformer (Vaswani et al., 2017) framework guided by memory modules. However, these methods primarily leverage transformers for the global modeling of time series, which results in the insufficient extraction of prototype features. In addition, the problem of storing prototype patterns of normal samples in the memory module directly impacts model detection performance. It is crucial to improve the updating process of memory items in the memory module to more accurately store prototype patterns of normal data.

To solve these challenges, we propose a memory-augmented state space model for multivariate time series anomaly detection, termed MemMambaAD. Specifically, we introduce a sequence decomposition state space model-temporal convolutional encoder, termed the sequence decomposition Mamba-TCN encoder, comprising a series decomposition module (Wu et al., 2021), a trend Mamba-TCN module, and a season Mamba-TCN module. The sequence decomposition Mamba-TCN encoder models the prototypical patterns of multivariate time series from both trend-season and global-local perspectives, thereby capturing the intrinsic patterns of multivariate time series at a higher resolution. Furthermore, the recursive mechanism of the Mamba block naturally accounts for the sequential nature of information, and the computational cost increases linearly with the amount of data, significantly reducing method parameters as well as memory consumption during training and testing. To enhance the storage of prototype patterns for normal samples in the memory module, we propose a Dynamic Memory Update Mechanism (DMUM). It dynamically updates memory items through the Memory Selection Mechanism (MSM), allowing for more rational updates to the memory items and achieving more accurate recording of the prototype patterns of normal samples. In practical anomaly detection in hydrogen energy scenarios, compared with existing methods, MemMambaAD alleviates the generalization of abnormal data, improves the detection effect of the method, reduces the memory resource consumption of the method, and better ensures production safety. Extensive experiments on five datasets demonstrate the effectiveness of our proposed method.

The contributions are summarized as follows:

- We propose a memory-augmented state space model for multivariate time series anomaly detection, termed MemMambaAD. Our method can fully extract prototype patterns and suppress the method's generalization ability on abnormal data, improving detection performance. Furthermore, MemMambaAD significantly reduces memory consumption while ensuring the accuracy of the method.
- We design a sequence decomposition Mamba-TCN encoder to solve the problem of insufficient extraction of prototype patterns of data. The encoder models the prototype patterns of multivariate time series from the perspective of trend-season and global-local, thereby capturing the intrinsic patterns of multivariate time series in a more fine-grained manner.
- We propose a dynamic memory update mechanism to address the issue of insufficient storage of normal prototype patterns in memory items. The memory items are flexibly adjusted through the memory selection mechanism to accurately record the prototype patterns of normal samples, to enhance the ability of the method to detect anomalies.

The rest of this paper is organized as follows. Section 2 introduces related work, Section 3 provides a preliminary to the related tasks and components, Section 4 outlines the working principle and details of MemMambaAD, Section 5 evaluates the performance of the proposed method, and we conclude in Section 6.

2. Related work

2.1. Unsupervised time series anomaly detection

As deep learning technology continues to advance, unsupervised time series anomaly detection has achieved significant success. DAGMM (Zong et al., 2018) integrates dimensionality reduction and density estimation within deep autoencoders for anomaly detection, achieving promising results. BeatGAN (Zhou et al., 2019) leverages the generative and discriminative mechanisms of generative adversarial networks to capture normal data patterns by generating and reconstructing time series data, thereby identifying anomalies. OmniAnomaly (Su et al., 2019) integrates the Variational Autoencoder (VAE) (Kingma and Welling, 2013) with recurrent neural networks (RNNs) in deep learning. This approach addresses complex multivariate time series, detecting anomalies by modeling their underlying distributions and temporal relationships. InterFusion (Li et al., 2021) employs two random latent variables alongside a VAE to model normal patterns in multidimensional time series data, deriving anomaly scores from reconstruction probabilities and ensuring model conciseness and efficiency. THOC (Shen et al., 2020) utilizes an RNN with skip connections to extract multi-scale features, which are subsequently fused through hierarchical clustering. EGNN (Guo et al., 2024) investigates multivariate correlations using a subgraph generation algorithm grounded in graph structure learning. It constructs a dependency graph comprising multiple subgraphs and their corresponding centers, subsequently leveraging subgraph center data for anomaly detection through prediction. A novel damage index approach (Entezami and Shariatmadar, 2018) and a combined method involving manifold learning, data clustering, and a non-parametric probabilistic model (Entezami et al., 2025) have been proposed to address environmental and operational variability, enhancing anomaly detection performance.

The attention mechanism (Vaswani et al., 2017) has been widely used by time series researchers due to its excellent full-process modeling capabilities. TranAD (Tuli et al., 2022) employs an adversarial training approach for the Transformer architecture, integrating it with meta-learning to enhance performance under limited training data conditions. DTAAD (Yu et al., 2024), built on Transformer and dual temporal convolutional networks, achieves outstanding performance by integrating autoregressive models with autoencoder structures for anomaly detection. AnomalyTrans (Xu et al., 2021) detects anomalies by modeling prior and sequence associations, computing their differences, and demonstrating robust performance across multiple datasets. DCdetector (Yang et al., 2023) employs a multi-scale dual-attention contrastive learning model, analyzing data points from two perspectives and distinguishing outliers through a difference-based criterion.

The core functionality of the TCN (Bai et al., 2018) lies in its use of causal convolution to capture local dependencies in time series data while ensuring no reliance on future information. TCN has found extensive applications in time series analysis. ModernTCN (Luo and Wang, 2024) enhances the traditional TCN network by decoupling depthwise separable convolutions, expanding its receptive field for time series data, and improving performance across various downstream time series tasks while preserving computational efficiency. M2N2 (Kim et al., 2024) introduces a test-time adaptation mechanism that adjusts model parameters during testing to better align with the normal distribution of test data.

Existing time series anomaly detection methods are mostly based on reconstruction ideas. Although they have achieved remarkable results

in this field, due to the construction of complex neural networks, such methods are susceptible to overfitting problems, thus limiting their detection performance. Different from the above methods, our method stores different prototype patterns through a memory module containing a dynamic memory update mechanism, and constrains the model through the normal data prototype pattern, so that it can focus more on the reconstruction of normal data, thereby effectively alleviating over-fitting problems and improve detection performance.

2.2. Applications of Mamba

Mamba (Gu and Dao, 2023) is a novel architecture that offers comparable modeling capabilities to the Transformer (Vaswani et al., 2017) while exhibiting lower computational complexity. Its introduction has garnered significant attention from researchers across diverse fields. Vision Mamba (Zhu et al., 2024) employs the bidirectional Mamba block as a general visual backbone, delivering superior performance and reduced computational complexity across various downstream tasks. U-Mamba (Ma et al., 2024) designed a hybrid CNN-SSM module and integrated it into a U-shaped framework for biomedical image segmentation tasks, achieving state-of-the-art performance. Mamba has also demonstrated significant potential in time series tasks. TimeMachine (Ahamed and Cheng, 2024) employs four Mamba blocks to design a dual-branch structure that integrates channel-independent and channel-mixed methods for time series prediction tasks, achieving state-of-the-art results. S-Mamba (Wang et al., 2024) employs bidirectional Mamba to capture variable correlations and utilizes a feedforward neural network to model temporal dependencies, achieving superior performance while maintaining efficiency. Bi-Mamba4TS (Liang et al., 2024) utilizes bidirectional Mamba to extract temporal dependencies from time series and introduces a mechanism that automatically selects between channel independence and channel mixing based on dataset characteristics, further advancing Mamba's application in time series prediction. Different from the above methods, this paper aims to investigate the performance of Mamba as a feature extraction backbone for time series anomaly detection.

2.3. Memory network

The core concept of memory networks (Weston et al., 2014) is to utilize external, readable, and writable memory to store and retrieve information, enabling the handling of complex tasks. Memory networks find widespread applications in the field of computer vision. MNAD (Park et al., 2020) introduces an encoder–decoder structure guided by a memory module for image anomaly detection and proposes an incremental update mechanism for the memory module. MPN (Lv et al., 2021) proposes a dynamic prototype unit that converts normal dynamics into prototypes in real time. The dynamic prototype unit updates during both the training and testing phases, reducing memory costs and enhancing detection performance. Inspired by the application of memory networks in computer vision, MemAugUTransAD (Qin et al., 2023) enhances anomaly detection performance through a memory network-guided U-transformer architecture. The model extracts multi-granular features from normal time series and employs multiple memory networks to store prototype patterns of normal data. MEMTO (Song et al., 2024) addresses the overfitting problem by designing a transformer structure guided by a memory module that adjusts memory updates based on input data. It also introduces a two-stage training paradigm for the memory module and anomaly score calculation criteria to enhance the anomaly detection performance. Different from these methods, this paper captures fine-grained time series feature prototypes from the perspectives of trend-season and global-local through the proposed sequence decomposition Mamba-TCN encoder. It enhances prototype quality in the memory module using high-quality features and introduces a dynamic memory update mechanism that adaptively updates memory items through the memory selection mechanism. The aim is to accurately record multiple prototype modes of normal samples, alleviating the overfitting phenomenon. Furthermore, our method reduces memory consumption during the training phase while improving anomaly detection performance.

3. Preliminaries

3.1. Problem description

Given a set of multivariate time series $X = \{x_1, x_2, x_3, \ldots, x_L\}$ of length L, each timestamp $x_l \in \mathrm{R}^N$ represents the data collected at time l, and N represents the dimension of the data. The goal of the multivariate time series anomaly detection task is to determine the abnormal result $Y_{test} = \{y_1, y_2, y_3, \ldots, y_L\}$, given a test time series X_{test} with the same dimension as the training data, where

$$y_t \in \begin{cases} 1, \text{ abnormal data point} \\ 0, \text{ normal data point} \end{cases}, \tag{1}$$

3.2. State space models

The state space model is a machine learning method used for time series prediction and modeling. Structured state space sequence models (S4) (Gu et al., 2021) is a variant, combining the advantages of recurrent neural networks and convolutional neural networks. The S4 model utilizes a structured state space to capture the characteristics of time series data. The state space consists of a set of hidden state variables that capture long-term dependencies in the data. Specifically, we define a state space, where h_t represents the hidden state at time t, encapsulating all relevant information of the time series data at that time. h_t is updated to the next time step through the state transition equation, and the observation y_t is derived from the current state h_t. The details are as follows:

$$h_{t+1} = Ah_t + Bu_t, \quad y_t = Ch_t, \tag{2}$$

where $A \in \mathrm{R}^{N\times N}$ represents the state transfer matrix, $B \in \mathrm{R}^{N\times D}$ represents the control input matrix, $C \in \mathrm{R}^{N\times D}$ represents the observation matrix, u_t represents the input, and y_t represents the output.

Secondly, it is converted into a discrete-time model through a discretization process. The details are as follows:

$$h_{k+1} = \alpha(\Delta t)h_k + \beta(\Delta t)u_k, \tag{3}$$

where $\alpha(\Delta t)$ and $\beta(\Delta t)$ are the discretized state transfer function and input function respectively, and Δt is the time step.

The S4 model utilizes convolution operations for parallel training, enhancing computational efficiency and effectively capturing the dynamic characteristics.

4. Method

4.1. Overall framework

As shown in Fig. 1, our method comprises a sequence decomposition Mamba-TCN encoder, a Dynamic Memory Update Mechanism (DMUM), and a decoder, with the sequence decomposition Mamba-TCN encoder consisting of a series decomposition module, a trend Mamba-TCN module, and a season Mamba-TCN module. First, the multivariate time series window $X = (x_l, x_{l+1}, \ldots, x_{l+t-1}) \in \mathrm{R}^{N\times t}$ (where each window consists of t consecutive time series data points) is fed into MemMambaAD as input. Subsequently, the encoder decomposes the window time series into trend terms and seasonal terms through the series decomposition module. The encoder independently extracts the trend and seasonal characteristics of the multivariate time series by designing the same trend Mamba-TCN module and season Mamba-TCN module, thereby capturing the intrinsic patterns of the time series more comprehensively. The DMUM is composed of two components:

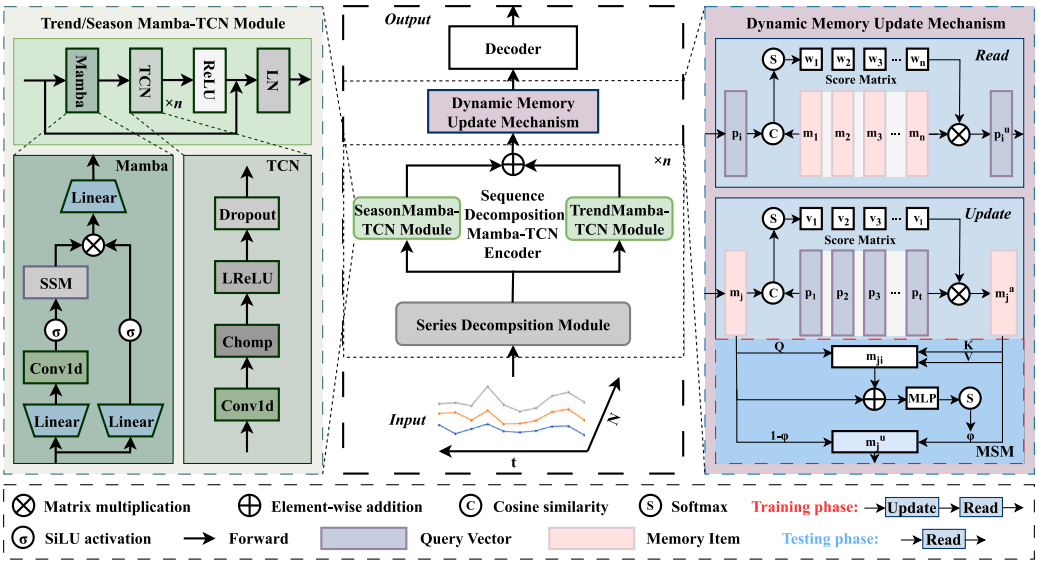

Fig. 1. Overall framework of MemMambaAD. The left picture shows the trend and season Mamba-TCN module structure, the middle picture shows the overall architecture of our method, and the right picture shows the Dynamic Memory Update Mechanism structure.

Read and *Update*. *Update* uses the query vector to update all memory items, while *Read* utilizes all memory items to enrich the features of the query vector. During the training phase, the output of the encoder serves as the query vector, and the memory items in the DMUM are dynamically adjusted through the Memory Selection Mechanism (MSM), enabling the DMUM to more accurately record the prototype patterns of normal samples and avoid generalizing to abnormal data. The query vector then retrieves the output from the memory module by reading the memory items. In the test phase, only the *Read* operation is executed to obtain the output of the memory module. The feature vector, enhanced by the memory module, is subsequently input into the decoder to reconstruct the sequence. The anomaly score is obtained by comparing the difference between the reconstructed sequence and the true sequence, and the score is compared with a threshold to determine whether the data point is anomalous. Algorithm 1 describes the overall mechanism of MemMambaAD. For clarity, we define the original time series as X and the reconstructed time series as $\hat{X}$.

Algorithm 1: Proposed Method MemMambaAD

Input: $X \in \mathbb{R}^{N\times t}$: time series window data
Output: $\hat{X} \in \mathbb{R}^{N\times t}$: reconstructed time series window data

// Sequence Decomposition Mamba-TCN Encoder
for $i = 1$ *to* n **do**
  $\text{trend}^i, \text{season}^i = \text{decomp}(X^i)$ *// Sequence decomposition*
  $\text{trend}^{i+1} = \text{ReLU}(\text{tcn}(\text{mamba}(\text{trend}^i))) + \text{trend}^i$
  $\text{season}^{i+1} = \text{ReLU}(\text{tcn}(\text{mamba}(\text{season}^i))) + \text{season}^i$
  $X^{i+1} = \text{LN}(\text{trend}^{i+1} + \text{season}^{i+1})$
// Dynamic Memory Update Mechanism
$p_i = \theta_{\text{enc}}(X)$ *// θ_{enc} : Encoder parameters*
if *training phase* **then**
  $m_j^a = \text{Update}(p_i, m_j)$ *//Dynamically update memory items*
  $p_i^u = \text{Read}(m_j, p_i)$ *//Update query vector*
else
  $p_i^u = \text{Read}(m_j, p_i)$
// Decoder
$\hat{X} = \theta_{\text{dec}}(p_1^u, \ldots, p_t^u)$ *// θ_{dec} : Decoder parameters*
return $\hat{X}$ *//Reconstructed time series window data*

4.2. Sequence decomposition Mamba-TCN encoder and decoder

The primary goal of the encoder is to comprehensively extract prototype features from the multivariate time series. Benefiting from the advantages of Transformer in long sequence modeling, existing methods usually tend to use Transformer alone for feature extraction of multivariate time series, ignoring the short-term patterns in time series. This limitation may result in the insufficient extraction of prototype patterns. In addition, the computational complexity of the global attention mechanism increases exponentially with the amount of data, significantly raising the demand for computational resources. To address these issues, we introduce a novel multi-view prototype extraction mechanism: the sequence decomposition Mamba-TCN encoder, as shown in Fig. 1(middle). The encoder significantly enhances anomaly detection performance by extracting fine-grained prototype features from trend-seasonal and global-local perspectives. The sequence decomposition Mamba-TCN encoder includes a series decomposition module, a trend Mamba-TCN module, and a season Mamba-TCN module. The series decomposition module decomposes the input time window into trend and seasonal components, enabling the independent extraction of fine-grained features. The trend Mamba-TCN and season Mamba-TCN modules are illustrated in Fig. 1(left). These modules employ the Mamba feature extractor to capture global time dependencies and the TCN feature extractor to capture local time dependencies, thereby fully capturing the intrinsic pattern of the time series. Compared with existing methods, our method significantly improves detection performance. For detailed experimental results, refer to Section 5.4. Next, we will provide a detailed introduction to the encoder.

Mamba Feature Extractor. Mamba introduces a selection mechanism that selects appropriate state transfer and observation matrices at different time steps, enhancing the method's ability to capture long-range time series features. Furthermore, Mamba exhibits linear time complexity and demonstrates great potential in processing long time series, making it suitable for extracting global features. As depicted in Fig. 1(left), let the input time series to the Mamba feature extraction block be $X \in R^{B\times N\times t}$, where B represents the data batch, N denotes the time series dimension, and t refers to the time series window length. Initially, two linear transformations are applied to obtain the dual variables X_1 and X_2. X_1 undergoes a convolution operation followed by nonlinear mapping using the SiLU activation function to obtain X_1', which is then

discretized using SSM to generate the state space representation y_1. X_2 is nonlinearly mapped via the SiLU activation function to obtain X_2', weighted by y_1, and then restored to its original dimension through linear mapping to obtain X'.

TCN Feature Extractor. TCN employs causal convolution to ensure the method relies solely on current and past input information for feature extraction, thereby effectively capturing the local features of data. As depicted in Fig. 1(left), the input time series is first processed with one-dimensional convolution to extract local features. To maintain the causality of the convolution operation and prevent leakage of future information, we apply suitable padding and clipping strategies to ensure the convolution depends solely on current and historical input data. Subsequently, the LeakyReLU activation function is applied for nonlinear activation on fine-grained time series features, effectively alleviating the gradient vanishing problem and enhancing the model's learning capability. Furthermore, LeakyReLU retains a certain activation value in the negative range, aiding in information retention and improving the robustness of the model. Simultaneously, dropout is applied to randomly deactivate some neurons, preventing overfitting and enhancing the model's generalization ability. TCN effectively captures short-term dependencies in time series by utilizing a shallow network structure.

Sequence Decomposition Mamba-TCN Encoder. We employ a multi-layer sequence decomposition Mamba-TCN encoder for feature extraction, where the output of each layer serves as the input to the subsequent layer. Specifically, when the input to the ith layer is X^i, the series decomposition module decomposes it into a trend term X^i_{trend} and a seasonal term X^i_{season}. The trend and seasonal terms are then passed to the trend Mamba-TCN module and the season Mamba-TCN module, respectively. Initially, the global time dependency is extracted through the Mamba block, resulting in a feature block containing rich global information. Next, the local fine-grained time dependency is captured through the multi-layer TCN block. Subsequently, nonlinear activation is applied using the ReLU function, and layer normalization is performed with the LayerNorm layer. The features are then continuously enriched via the residual connection. Finally, the trend and seasonal term features are combined to obtain the new time series feature X^{i+1}. X^{i+1} serves as the input to the next layer, and the final output of the sequence decomposition Mamba-TCN encoder is X^s. The details are as follows:

$$X^i_{\text{trend}} = AvgPool\left(Padding\left(X^i\right)\right), \tag{4}$$

$$X^i_{\text{season}} = X^i - X^i_{\text{trend}}, \tag{5}$$

$$X^{i+1}_{\text{trend}} = LN\left(\xi\left(TCN\left(Mamba\left(X^i_{\text{trend}}\right)\right)\right) + X^i_{\text{trend}}\right), \tag{6}$$

$$X^{i+1}_{\text{season}} = LN\left(\xi\left(TCN\left(Mamba\left(X^i_{\text{season}}\right)\right)\right) + X^i_{\text{season}}\right), \tag{7}$$

$$X^{i+1} = X^{i+1}_{\text{trend}} + X^{i+1}_{\text{season}}, \tag{8}$$

where $AvgPool$ represents the moving average, $Padding$ represents the padding operation on the sequence to keep the length unchanged, ξ represents the ReLU activation function and LN represents the LayerNorm layer normalization operation.

Decoder. For the decoder, we define it as a linear mapping layer, reconstructing the feature vector output by the dynamic memory update mechanism to obtain $\hat{X}$. The details are as follows:

$$\hat{X} = P\phi + b, \tag{9}$$

where P represents the output of the dynamic memory update mechanism, ϕ represents the weight matrix, and b represents the bias matrix.

4.3. Dynamic memory update mechanism

The purpose of the Memory module is to accurately record the prototype patterns of normal samples, thereby preventing the model from generalizing to anomalous samples. Existing memory modules typically employ incremental methods to update memory items. However, the straightforward incremental update method may result in the inadequate storage of prototype patterns of normal samples within memory items. We have enhanced this approach. The DMUM adaptively updates memory items via the MSM. This allows for more effective updates of memory items, thereby more accurately capturing the prototype patterns of normal samples, which improves anomaly detection performance. The encoder produces the output $p = \left(p_1, p_2, \ldots, p_t\right) \in \mathrm{R}^{N\times t}$, where $p_i \in \mathrm{R}^N$ (for $i = 1, 2, \ldots, t$) represents the query vector at time step i. The DMUM consists of n memory items to store the prototype patterns of normal samples. We denote the memory items in the DMUM as $m_j \in \mathrm{R}^N$ (for $j = 1, 2, \ldots, n$). The DMUM primarily consists of the $Read$ and $Update$ operations, with the MSM incorporated within the $Update$ phase. In the training phase, the query vector first passes through the $Read$ operation to generate a new feature vector, and then the memory item is updated through the $Update$ operation. In the testing phase, the query vector only passes through $Update$. DMUM is described as follows.

Read: As shown in Fig. 1(right), given a query vector p_i, we calculate the cosine similarity between p_i and all memory items m_j to obtain the correlation coefficient matrix between p_i and all memory items, and then normalize the matrix to obtain the score matrix. This score matrix is used to update the query vector p_i. For each query vector p_i, we perform matrix multiplication of the score matrix and the memory item to obtain the query output, which is more aligned with the prototype pattern of normal data while preserving its original features. Finally, we concatenate the query vector p_i and the query output p^u_i to form a new feature vector, which is subsequently sent to the decoder. The details are as follows:

$$W_n = softmax\left(\frac{p_i m_j^T}{\|p_i\|\left\|m_j\right\|}\right), \tag{10}$$

$$p^u_i = \sum_{j=1}^{n} w_j m_j, \tag{11}$$

$$\hat{p}_i = concat\left(p_i, p^u_i\right), \tag{12}$$

where W_n represents the score matrix, w_j represents the weight of the memory item, p^u_i represents the query vector after reading the memory item, and $\hat{p}_i$ represents the concatenated feature vector.

Update: As shown in Fig. 1(right), given a memory item m_j, we calculate the cosine similarity between m_j and all query vectors p_i to obtain the relationship coefficient matrix between m_j and all query vectors, and then normalize the matrix to obtain the score matrix. The score matrix is used to select and update the memory items in the memory module. For each memory item m_j, we multiply the score matrix with the query vector to obtain the temporary memory increment. The details are as follows:

$$V_l = softmax\left(\frac{m_j p_i^T}{\left\|m_j\right\|\|p_i\|}\right), \tag{13}$$

$$m^a_j = \sum_{i=1}^{l} v_i p_i, \tag{14}$$

where V_l represents the score matrix, v_i represents the weight of the ith query vector, and m^a_j represents the temporary memory increment.

The MSM aims to update the memory items in the memory module moderately and adaptively to enable it to store diverse normal prototype features, thereby improving the method's detection performance. We use the cross-attention mechanism to enhance the prototype pattern

in memory items using the temporary memory increment and derive the temporary memory item. Specifically, a linear mapping is performed on the memory item m_j to obtain Q, while two linear mappings are applied to the temporary memory increment m_j^a to obtain K and V. The query matrix Q is multiplied by K, and the score matrix is obtained after normalization, and finally multiplied by the matrix V to obtain the temporary memory item. To further enrich the prototype pattern in the memory item, the temporary memory item is added to the temporary memory increment, enabling autonomous learning via the MLP structure. The MLP first maps it to a high-dimensional space, then performs nonlinear mapping through the ReLU activation function, and finally projects it back to the original dimensionality. After normalization, the memory selection threshold is obtained. The temporary memory item and the temporary memory increment are adaptively weighted through the memory selection threshold to obtain the updated memory item. MSM controls the extent to which the temporary memory increment m_j^a updates the memory item m_j, as well as the extent to which m_j decays. This adaptive update method and the ability to store multiple normal prototype patterns in the item help increase the performance of the method. The details are as follows:

$$Q, K, V = W_1 m_j, W_2 m_j^a, W_3 m_j^a, \tag{15}$$

$$m_{ji} = \left(softmax\left(\frac{QK^T}{\sqrt{d}} \right) \right) V, \tag{16}$$

$$\varphi = softmax\left(MLP\left(m_{ji} + m_j^a \right) \right), \tag{17}$$

$$m_j^u = \varphi \odot m_j^a + (1 - \varphi) \odot m_{ji}, \tag{18}$$

where m_j represents the memory item, m_j^a represents the temporary memory increment, W_1, W_2, W_3 represents the learnable mapping matrix, and m_{ji} represents the temporary memory item, d is 256, $\odot$ represents element-wise multiplication, MLP consists of linear mapping and ReLU activation function, φ represents the memory selection threshold, and m_j^u represents the updated memory item.

4.4. Training setting

Our method adopts the training framework and loss function of Song et al. (2024). In the first phase, MemMambaAD is trained by reconstructing the input, and the trained encoder generates query vectors for randomly sampled 10% of the training data. We then apply the K-means clustering algorithm to cluster the queries and assign each centroid as the initial value of the memory item. In the second stage, MemMambaAD uses these initialized memory items for training on the anomaly detection task. In the training phase, the original time series data is defined as X, the reconstructed time series data is defined as $\hat{X}$, and the weight of the memory item corresponding to the query vector is defined as w_j. The loss function of the method is defined as follows:

Reconstruction loss. We minimize the reconstruction loss during the training phase, which is defined as the $L2$ loss between X and $\hat{X}$:

$$\mathcal{L}_{\text{rec}} = \|X - \hat{X}\|_2^2, \tag{19}$$

Sparse loss. To alleviate the over-reconstruction of abnormal samples by dense memory terms, a Sparse loss (Gong et al., 2019) is used to minimize the entropy of W_n:

$$\mathcal{L}_{\text{spar}} = \sum_{j=1}^{n} -w_j \odot \log\left(w_j\right), \tag{20}$$

Final loss. The final objective function of this paper is presented below:

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \vartheta \mathcal{L}_{\text{spar}}, \tag{21}$$

where ϑ represents the weighting coefficient, which is taken as 0.01 in this paper.

Anomaly score. In the inference phase, to further amplify the difference in scores between normal data and abnormal data, we use the difference between the query vector and the memory item closest to it as a weight to weight the error between the reconstructed value and the true value to calculate the anomaly score. Specifically, the anomaly score of timestamp i is defined as follows:

$$Score\left(X_i\right) = softmax\left(\left\| p_i - \widetilde{m}_j \right\|_2^2 \right) \odot \left\| X_i - \hat{X}_i \right\|_2^2. \tag{22}$$

where p_i represents the query vector corresponding to X_i, $\widetilde{m}_j$ represents the memory item closest to p_i, $\odot$ represents element-wise multiplication, and $\|\|_2^2$ represents mean square error.

Table 1
Details of benchmark datasets.

Benchmark	Window	Dimension	Train	Test	Anomalies (%)
MSL	100	55	46 653	73 729	10.5
SMAP	100	25	108 146	427 617	12.8
PSM	100	25	105 984	87 841	27.8
SWaT	100	51	396 000	449 919	12.0
NIPS-TS-GECCO	100	9	69 260	69 261	1.1

5. Experiments

5.1. Datasets

To fully verify and evaluate our method, we selected five public benchmark datasets. These datasets cover a variety of application scenarios of time series anomaly detection tasks, including aerospace, network servers, and industrial fields, which can fully demonstrate the applicability and performance advantages of the method. Among them, the NIPS-TS-GECCO exhibits significant anomaly sparsity. Through experiments on these datasets, we demonstrate the versatility, robustness, and practical utility of the method in real-world multi-domain time series anomaly detection tasks. They are introduced as follows: (1) Mars Science Laboratory dataset (MSL): data collected by the NASA Mars rover, covering multiple entities, each entity monitored by 55 variables (Nakamura et al., 2020). (2) Soil Moisture Active Passive dataset (SMAP): soil samples and telemetry information using the Mars rover by NASA (Hundman et al., 2018). (3) Pooled Server Metrics dataset (PSM): collected internally from multiple application server nodes at eBay with 25 dimensions (Abdulaal et al., 2021). (4) Secure Water Treatment dataset (SWaT): collected from a real-world water treatment plant with 7 days of normal and 4 days of abnormal operation (Mathur and Tippenhauer, 2016). (5) NIPS-TS-GECCO: comes from the NIPS Time Series Anomaly Detection Competition and the GECCO Challenge, and contains time series data in a variety of real scenarios and synthetic environments (Lai et al., 2021) (see Table 1).

5.2. Baselines and evaluation metric

We compare our method with 13 state-of-the-art methods to evaluate its performance, including DAGMM (Zong et al., 2018), DeepSVDD (Ruff et al., 2018), LSTM-VAE (Park et al., 2018), BeatGAN (Zhou et al., 2019), OmniAnomaly (Su et al., 2019), InterFusion (Li et al., 2021), THOC (Shen et al., 2020), EGNN (Guo et al., 2024), AnomalyTrans (Xu et al., 2021), MEMTO (Song et al., 2024), DCdetector (Yang et al., 2023), ModernTCN (Luo and Wang, 2024), and M2N2 (Kim et al., 2024).

To achieve a comprehensive understanding of the method's performance, we use three indicators for method evaluation: precision (P), recall (R), and F1 score (F1). Specifically, the F1 score is the weighted harmonic average of precision and recall. It provides a comprehensive measure of the method's accuracy and recall, effectively reflecting its overall performance in practical scenarios. The details are as follows:

$$P = \frac{TP}{TP + FP}, \tag{23}$$

Table 2
Performance comparison of MemMambaAD with baseline methods on four real-world datasets. P: Precision, R: Recall, F1: F1 score. The best results are in bold.

Method	MSL			SMAP			PSM			SWaT			Avg.
	P↑	R↑	F1↑	P↑	R↑	F1↑	P↑	R↑	F1↑	P↑	R↑	F1↑	F1↑
DAGMM	89.60	63.93	74.62	86.45	56.73	68.51	93.49	70.03	80.08	89.92	57.84	70.40	73.40
Deep-SVDD	91.92	76.63	83.58	89.93	56.02	69.04	95.41	86.49	90.73	80.42	84.45	82.39	81.43
LSTM-VAE	85.49	79.94	82.62	92.20	67.75	78.10	73.62	89.92	80.96	76.00	89.50	82.20	80.97
BeatGAN	89.75	85.42	87.53	92.38	55.85	69.61	90.30	93.84	92.04	64.01	87.46	73.92	80.77
OmniAnomaly	89.02	86.37	87.67	92.49	81.99	86.92	88.39	74.46	80.83	81.42	84.30	82.83	84.56
InterFusion	81.28	92.70	86.62	89.77	88.52	89.14	83.61	83.45	83.52	80.59	85.58	83.01	85.57
THOC	88.45	90.97	89.69	92.06	89.34	90.68	88.14	90.99	89.54	83.94	86.36	85.13	88.76
EGNN	77.20	94.38	84.92	88.29	90.14	89.21	93.29	88.91	91.05	**99.28**	63.33	77.33	85.63
AnomalyTrans	91.77	95.54	93.62	93.58	98.87	96.15	95.75	98.4	97.05	91.55	93.35	92.44	94.81
MEMTO	91.77	94.31	93.02	**93.78**	99.24	96.43	97.44	98.33	97.88	90.35	99.82	94.85	95.54
DCdetector	**92.22**	96.45	94.29	93.48	99.01	96.16	97.19	97.42	97.31	93.34	99.15	96.16	95.98
ModernTCN	89.69	75.04	81.87	89.94	55.16	68.38	98.78	93.02	95.81	95.87	89.57	92.61	84.66
M2N2	–	–	94.40	–	–	70.80	–	–	–	–	–	90.30	–
MemMambaAD	92.06	**98.07**	**94.97**	93.61	**99.48**	**96.46**	**97.51**	**99.26**	**98.38**	93.24	**100**	**96.50**	**96.57**

$$R = \frac{TP}{TP + FN}, \tag{24}$$

$$F1 = \frac{2 \times P \times R}{P + R}. \tag{25}$$

In addition, we employ the Range-AUC-PR (R_AUC_PR) and Range-AUC-ROC (R_AUC_ROC), as well as the Volume Under the Surface-PR (VUS_PR) and Volume Under the Surface-ROC (VUS_ROC) (Paparrizos et al., 2022), to perform a deeper evaluation of the proposed method. The R_AUC_PR and R_AUC_ROC offer enhanced reliability by surmounting the limitations of threshold dependency and effectively accommodating range-based anomalies. Meanwhile, the VUS_PR and VUS_ROC facilitate a parameter-independent assessment, thereby addressing challenges such as temporal delays and noise. This integrative approach provides a more precise and holistic evaluation of the performance of our proposed method.

5.3. Implementation details

We introduce all the parameter configurations in the experiment below. Our sliding window length is 100, the batch size for NIPS-TS-GECCO is set to 128, and the batch size for the other datasets is 256, and the Adam optimizer is used to train MemMambaAD. The learning rate for the first training is 0.0001, and the learning rate for the second training is 0.00005. All datasets are trained for 10 epochs. MemMambaAD contains 3 layers of sequence decomposition Mamba-TCN encoder, and the trend Mamba-TCN module and season Mamba-TCN module contain 3 layers of TCN. The number of memory items in the dynamically updated memory module is 10, the number of heads in the cross-attention module is 1, and the hidden dimension is 256. The anomaly threshold $\tau = 1$ for all datasets, and the loss coefficient ϑ is 0.01. In addition, we use Python version 3.10 and Pytorch version 2.1. All experiments are conducted on an NVIDIA GeForce RTX 24G 3090 GPU.

5.4. Comparison with state-of-the-art methods

Quantitative Evaluation. Table 2 presents the experimental outcomes of our approach in comparison with 13 deep learning methods on four publicly available datasets. Among them, the experimental data of AnomalyTrans, MEMTO, DCdetector, and ModernTCN are reproduced according to their source code, the data of EGNN and M2N2 are cited from the original paper, and the rest of the data are from Xu et al. (2021). The results indicate that the proposed MemMambaAD achieves the best recall and F1 score on all datasets, with an average F1 score of 96.57%. While the precision metric is not the highest, it still ranks in the top three on multiple datasets. This could be explained by the fact that the distribution of time series data changes over time, resulting in the method's inability to consistently identify normal data. The results show that the detection performance of Transformer-based methods is generally better than that of Variational AutoEncoders or Convolution-based methods. For example, DCdetector and AnomalyTrans have better F1 scores than LSTM-VAE and ModernTCN. This is due to the transformer's ability to model long sequences globally and extract temporal dependencies effectively. Compared with AnomalyTrans, our method demonstrates a significant advantage in accuracy, recall, and F1 score, with an average F1 score improvement of 1.76%. Our sequence decomposition Mamba-TCN encoder models the prototype features of time series from multiple perspectives: trend-season and global-local, extracting global dependencies while focusing on local details, surpassing the sequence association method of AnomalyTrans. The DCdetector based on the contrastive learning method uses multi-scale patches to extract features at multiple scales and uses a dual attention mechanism to model both intra-patch and inter-patch, achieving leading detection results. Compared with it, our method demonstrates superiority in recall rate and F1 score, increasing the average F1 score from 95.98% to 96.57%. This improvement is attributed to our method's use of a memory module to store prototype features of normal data, which moderately limits the model's generalization to abnormal samples, thereby enhancing anomaly detection performance. ModernTCN, an innovative approach to traditional convolution, has achieved good results in multiple time series tasks. However, MemMambaAD outperformed it in F1 by 11.91%. The latest method, M2N2, employs test-time adaptation to adjust normal data distributions during testing, enhancing the method's detection performance. In comparison, MemMambaAD uses a dynamic memory module to store normal prototypes, achieving an overall lead in F1 scores. MEMTO is the most advanced anomaly detection method based on memory modules. In comparison, our method improves the F1 score from 95.54% to 96.57%, highlighting our method's ability to extract time series prototype features more effectively. Additionally, the dynamic memory update mechanism enables more accurate prototype storage, fully verifying the effectiveness of our approach.

Table 3 shows the experimental results of each method on the four indicators of R_AUC_PR, R_AUC_ROC, VUS_PR, and VUS_ROC. The results indicate that MemMambaAD demonstrates excellent performance on all datasets, particularly on the MSL, SMAP, and PSM datasets, where it exhibits clear superiority in all four metrics. This highlights the method's reduced sensitivity to threshold selection and its robustness against challenges such as data noise, time delays, and changes in abnormal proportions. On the SWaT dataset, while some metrics of MemMambaAD were not the highest, the method still demonstrated competitiveness. Overall, MemMambaAD achieved the highest average scores across all four metrics on the four datasets, surpassing the second-best method by 1.54%, 2.02%, 2.53%, and 1.60%, respectively. These results strongly validate the effectiveness and broad applicability of MemMambaAD in time series anomaly detection tasks.

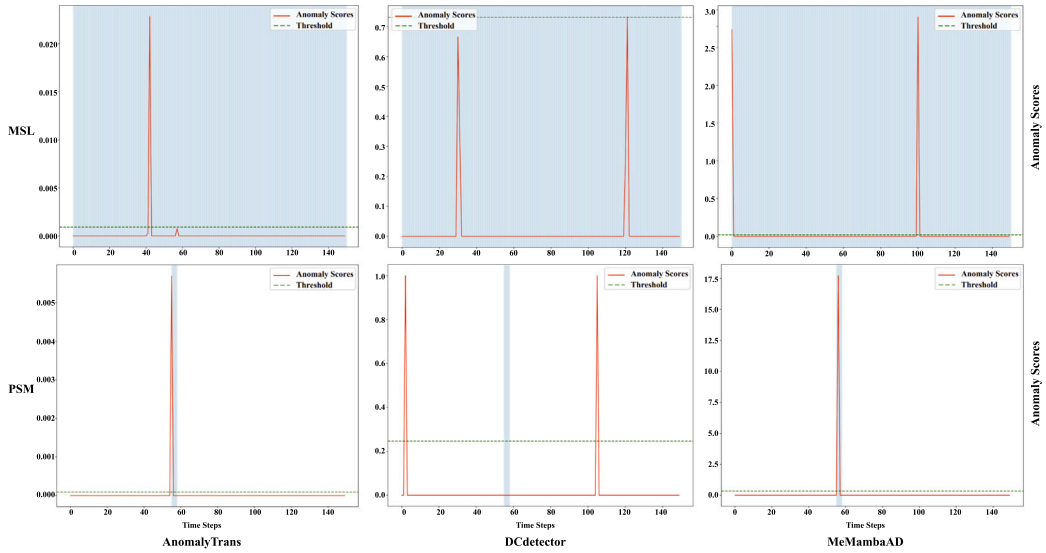

Fig. 2. Qualitative comparison with AnomalyTrans and DCdetector on MSL and PSM dataset. In the figure, the red curve represents the anomaly score, the green dotted line represents the anomaly threshold, and the blue shadow represents the true anomaly.

Table 3
Performance comparison of MemMambaAD with baseline methods on four real-world datasets. The best results are in bold.

Dataset	Method	R_AUC_PR↑	R_AUC_ROC↑	VUS_PR↑	VUS_ROC↑
MSL	AnomalyTrans	87.65	89.84	86.28	88.27
	MEMTO	86.20	88.22	85.92	87.88
	DCdetector	92.56	91.03	88.88	86.09
	MemMambaAD	**95.74**	**95.10**	**93.84**	**92.42**
SMAP	AnomalyTrans	93.67	95.85	92.69	94.72
	MEMTO	93.90	96.03	93.45	95.51
	DC	94.32	96.65	93.17	95.33
	MemMambaAD	**94.54**	**96.84**	**93.78**	**95.96**
PSM	AnomalyTrans	93.46	92.62	91.54	89.94
	MEMTO	93.48	91.88	91.71	89.43
	DCdetector	92.56	91.03	88.88	86.09
	MemMambaAD	**95.74**	**95.10**	**93.84**	**92.42**
SWaT	AnomalyTrans	84.64	86.66	85.33	87.44
	MEMTO	**94.01**	**97.90**	93.92	**97.79**
	DCdetector	93.95	96.36	**93.99**	96.41
	MemMambaAD	93.54	96.11	93.67	96.23

To further enhance the comprehensiveness and depth of method evaluation, we conducted a thorough evaluation of MemMambaAD using the NIPS-TS-GECCO dataset, characterized by an extremely low anomaly rate.

Table 4 presents a performance comparison between MemMambaAD and other leading methods on the NIPS-TS-GECCO dataset. The data demonstrate that our method achieves significant advantages in all evaluation metrics. Specifically, our method achieves an F1 score of 55.94%, which is 20.8% higher than the best performing baseline method, indicating superior anomaly detection performance. In addition, on the range curve indicators (R_A_R and R_A_P), our method achieved 67.22% and 40.98%, respectively, indicating superior range detection capabilities compared to other methods. In terms of volumetric curve metrics (V_ROC and V_PR), our method achieves 66.91% and 40.71%, respectively, which are 5.30% and 18.70% higher than the best baseline method. In summary, our method still performs well even in datasets with extremely sparseness.

Qualitative Evaluation. Fig. 2 provides the experimental results of qualitative comparison. We collected data with a length of 150 timestamps from the test set and showed the anomaly scores of AnomalyTrans, DCdetector, and our method under the point adjustment strategy. On the MSL dataset, it is evident that AnomalyTrans detected only a single anomaly, while the anomaly scores produced by DCdetector remained below the defined threshold. In contrast, MemMambaAD demonstrated greater sensitivity to anomalies, with its anomaly scores significantly exceeding the threshold. On the PSM dataset, although AnomalyTrans detected anomalies, its localization was imprecise. Additionally, DCdetector failed to accurately identify anomalies. However, MemMambaAD not only detected anomalies but also precisely localized them. Among the evaluated methods, the detection results of our approach most closely aligned with the ground truth, highlighting its superiority in addressing complex anomaly scenarios.

GPU Memory Usage Analysis. MemMambaAD improves anomaly detection performance while significantly reducing the memory overhead of memory module-based methods. Benefiting from Mamba's ability to sequentially encode data, our method omits the step of positional encoding and directly maps time series data into the feature space. Consequently, the method reduces the memory term's parameter count without compromising performance, thereby minimizing memory usage. Additionally, the Mamba model minimizes memory usage by leveraging mixed-precision training, dynamic computation graphs, and efficient memory management strategies. These technologies allow the model to load only necessary parameters during calculation, reducing memory consumption while maintaining efficient training and inference performance. Therefore, our method achieves enhanced accuracy while significantly reducing memory usage. Fig. 3(a) compares the GPU memory consumption of our method to that of the baseline. The blue color indicates GPU memory usage during training, and the orange color indicates usage during testing. Our results demonstrate that our method reduces GPU memory usage in the training phase by 77.61%, 44.9%, 79.27%, and 67.15% across four datasets, with an average reduction of 68.64%. During the testing phase, GPU memory usage is reduced by 41.51%, 65.17%, 65.40%, and 44.92% on the four datasets, averaging a reduction of 54.23%. This approach significantly reduces memory usage while maintaining high performance, enhancing the algorithm's resource efficiency and scalability.

To further verify the impact of sequence decomposition Mamba-TCN encoder and Dynamic Memory Update Mechanism (DUMU) on the time and space complexity of the method. We further analyzed the number of method parameters, average GPU usage during dataset training and testing, and average F1 score. As shown in Fig. 3(b), when we use DMUM to replace the memory module, although the number of parameters and GPU consumption increase slightly, our average F1

Table 4
Performance comparison of MemMambaAD with baseline methods on NIPS-TS-GECCO. The best results are in bold.

Method	NIPS-TS-GECCO						
	P↑	R↑	F1↑	R_AUC_PR↑	R_AUC_ROC↑	VUS_PR↑	VUS_ROC↑
AnomalyTrans	30.98	51.78	38.77	29.44	61.44	28.76	60.78
MEMTO	37.21	60.96	46.21	33.90	63.18	34.31	63.61
DCdetector	37.65	56.58	45.21	33.38	62.36	32.64	61.65
MemMambaAD	**43.88**	**77.12**	**55.94**	**40.98**	**67.22**	**40.71**	**66.91**

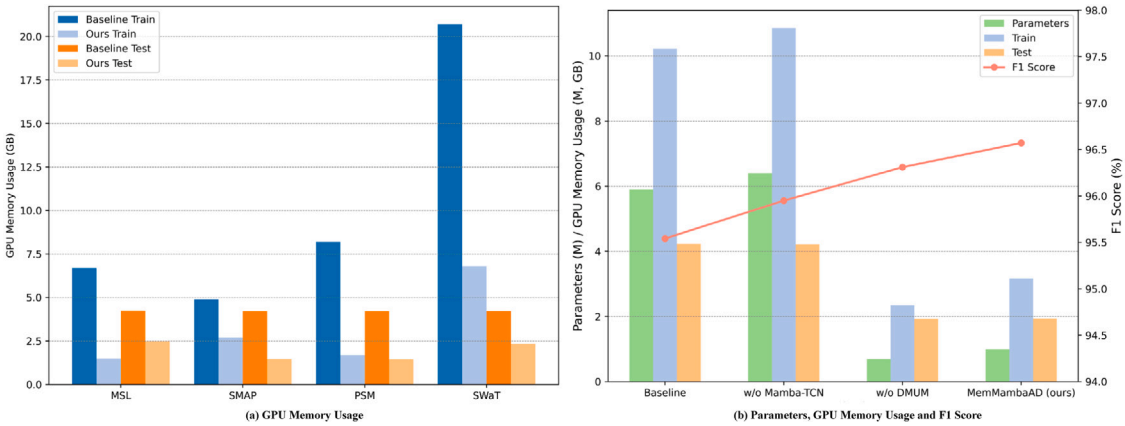

Fig. 3. (a): Comparison of GPU memory cost of MemMambaAD and Baseline. (b): Impact of sequence decomposition Mamba-TCN encoder and Dynamic Memory Update Mechanism (DMUM) on model parameters, average GPU usage, and average F1 score.

score is improved. We believe that the small increase in parameters is acceptable. When we use the sequence decomposition Mamba-TCN encoder instead of the Transformer, the number of parameters of the method is reduced from 5.9M to 0.7M, the GPU consumption is significantly reduced, and the F1 score is also improved. When the sequence decomposition Mamba-TCN encoder is used together with DMUM, the number of parameters of MemMambaAD is only 16.95% of the baseline, the GPU consumption for training and testing is only 31.02% and 45.96% of the baseline, and the highest F1 score is achieved. This fully demonstrates that MemMambaAD improves the accuracy of the method while taking into account efficiency.

5.5. Ablation study

To further evaluate the importance of each module in our method, we remove or substitute alternatives to observe their effect on the average F1 score. Specifically, we replace the sequence decomposition Mamba-TCN encoder with a Transformer and remove the Memory Selection Mechanism (MSM) to assess the impact on method performance. Table 5 shows our experimental results. We found that replacing the sequence decomposition Mamba-TCN encoder with Transformer results in a 0.62% drop in the average F1 score, which indicates that more fine-grained temporal features can be extracted from the perspectives of trend-season and global-local, thus promoting the quality of prototypes stored in memory items and helping to improve anomaly detection performance. After removing the MSM, the average F1 score dropped by 0.26%. This shows that our MSM can more reasonably integrate memory items and memory increments through adaptive weight allocation so that the memory items can more accurately record the prototype patterns of normal data and improve the method's anomaly detection ability. When we use the sequence decomposition Mamba-TCN encoder and the MSM at the same time, our method improves the F1 score by 1.03% compared with the baseline, which shows that our method can extract prototype features more fully. The fine-grained prototype features, in combination with the MSM, enhance the dynamic memory update mechanism, enabling more accurate storage of normal data prototype patterns, thus effectively improving anomaly detection performance.

Table 5
Ablation studies in MemMambaAD. The best results are in bold.

Method	F1↑				
	MSL	SMAP	PSM	SWaT	Avg.
Baseline	93.02	96.43	97.88	94.85	95.54
w/o Mamba-TCN	94.44	96.46	97.91	95.02	95.95
w/o MSM	94.29	96.46	98.05	96.45	96.31
MemMambaAD(ours)	**94.97**	**96.46**	**98.38**	**96.50**	**96.57**

Table 6
Ablation studies in sequence decomposition Mamba-TCN encoder. The best results are in bold.

Method	F1↑				
	MSL	SMAP	PSM	SWaT	Avg.
w/o Decomp	94.97	96.36	98.11	96.46	96.47
w/o Mamba	94.23	**96.65**	98.00	96.28	96.29
w/o TCN	94.50	96.22	98.09	96.45	96.31
MemMambaAD(ours)	**94.97**	96.46	**98.38**	**96.5**	**96.57**

Table 7
Ablation studies in dynamic memory update mechanism. The best results are in bold.

Method	F1↑				
	MSL	SMAP	PSM	SWaT	Avg.
Add	93.13	96.04	97.76	96.42	95.83
Gate	94.29	96.46	98.05	96.45	96.31
UR	94.97	96.45	98.10	96.35	96.46
MemMambaAD(ours)	**94.97**	**96.46**	**98.38**	**96.5**	**96.57**

In addition, we further explore the role of the components in the sequence decomposition Mamba-TCN encoder. We remove the series decomposition module, Mamba block, and TCN block to observe their impact on the average F1 score. Table 6 shows our experimental results. When we removed the series decomposition module, the F1 score dropped by 0.1%, indicating that time series decomposition is more convenient for extracting prototype data features from the perspective of trends and seasons. When we removed the Mamba block and the TCN block, the F1 scores dropped by 0.28% and 0.26%, respectively,

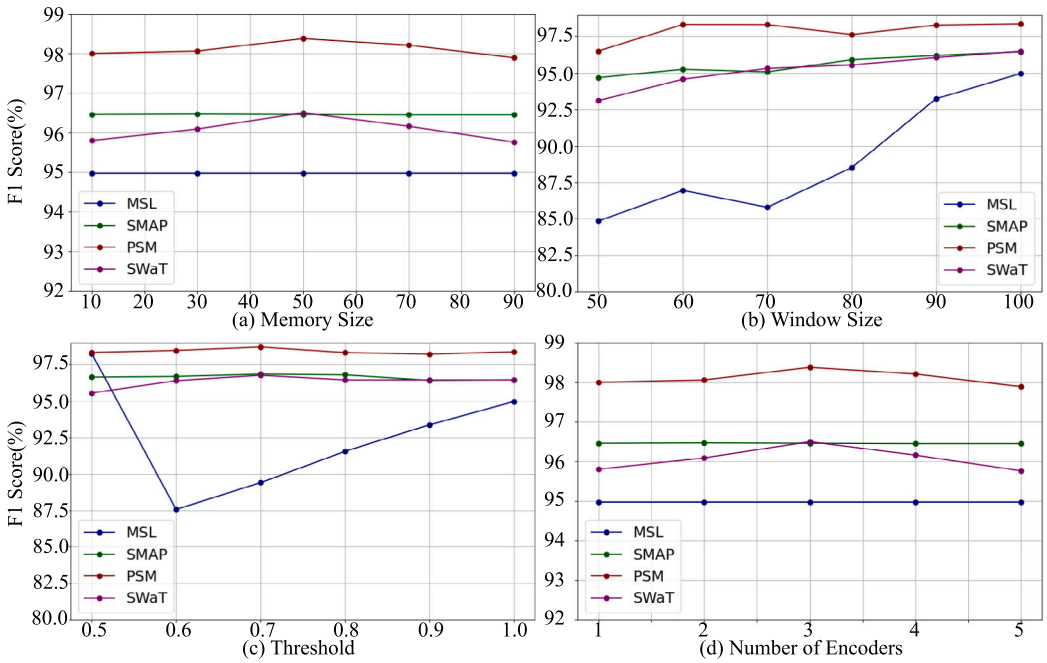

Fig. 4. Parameter sensitivity studies of main hyper-parameters in MemMambaAD.

indicating that it is difficult to fully model the feature prototype from both a local and a global perspective. Feature extraction from a global perspective will lack control over local details, and feature extraction from a local perspective will ignore the long-term characteristics of the time series. When the module combines the series decomposition module with the Mamba block and the TCN block, it extracts prototype patterns from multiple perspectives, including trend-season and global-local, focusing on both the long-term characteristics of the time series and the capture of local fine-grained features. This enhances the method's ability to extract prototype features from normal data, thereby improving its ability to detect anomalies.

We further investigate the role of the DMUM and adopt two solutions to replace the key component MSM in $Update$. On the one hand, we use the incremental update method to directly add the addition to the memory item, recorded as Add, and on the other hand, we use the update strategy in Song et al. (2024), recorded as Gate. In addition, we further explore the impact of the placement of update and read on the model effect. UR means updating the memory item first and then performing the read operation. Table 7 shows our experimental results. Compared with Add and Gate, MSM achieved an F1 score lead of 0.74% and 0.26% respectively. Compared with the UR method, DMUM achieved an F1 score lead of 0.11%. These results demonstrate the effectiveness of MSM in dynamically updating memory items. DMUM more effectively records the prototype pattern of normal samples, suppresses the generalization ability of the method to abnormal samples, and enhances the anomaly detection performance of the method.

5.6. Parameter sensitivity

We explore how the performance of the method is affected by changes in hyperparameters. The resource usage of the method will increase as the number of memory items n increases. How to balance method performance and method complexity is a key issue. Based on the common settings of existing memory-based methods, we explore the performance of our method when the number of memory items is $n \in \{10, 30, 50, 70, 90\}$. As shown in Fig. 4(a), the sensitivity analysis results for the memory items in the memory module are shown. The results show that for changes in memory items, the F1 scores of the MSL, SMAP, and SWaT datasets only change slightly, and the F1 score of the PSM dataset fluctuates by 0.47%. When the number of memory items $n = 10$, the method performance is the best, and the method takes into account both performance and complexity. The above results show that our method has good robustness to the number of memory items.

Time series sliding windows are widely used in various time series tasks. The size of the sliding window is an important hyperparameter. A window that is too small may cause the model to fail to capture enough contextual information, while a window that is too large may increase computational complexity or even introduce noise. We chose a window size range of 50 to 100 for exploration, based on the common settings in existing methods (existing time series models use windows of similar sizes). As shown in Fig. 4(b), we set the time series sliding window to $t \in \{50, 60, 70, 80, 90, 100\}$. The results show that the MSL dataset is sensitive to the sliding window size, and the F1 score generally improves as the window increases. This may be because its time series has a long periodicity, the features are easy to capture in long sequences, and it is more difficult to model short sequences. The F1 scores of the remaining datasets fluctuate slightly with the change of the sliding window, showing good robustness. The window size of all datasets in this paper is set to 100.

The anomaly threshold is a key parameter for distinguishing normal from abnormal data, and its setting is crucial to the accuracy and robustness of the method. According to existing literature, the anomaly threshold is usually set between 0.5 and 1, which is a common range and suitable for a variety of anomaly detection tasks. We explored this range to evaluate the impact of threshold changes on the method detection performance. The results are shown in Fig. 4(c). The F1 score of the MSL dataset fluctuates greatly with the change in the anomaly threshold. The other three datasets show strong robustness and achieve good F1 scores under multiple threshold conditions. The anomaly thresholds of all datasets in this paper are set to 1, which may not be optimal. Future research can explore more sophisticated threshold adjustment strategies.

We vary the number of layers in the sequence decomposition Mamba-TCN encoder to explore its impact on the performance of our method. Specifically, we varied the number of layers in the encoder from 1 to 5 and observed the changes in the method's F1 score on the dataset. The results are presented in Fig. 4(d). The experiment demonstrates that the performance fluctuation range of MemMambaAD

is 0.31%. When the number of layers is 3, the method's average F1 score reaches its peak at 96.57%. This result demonstrates that our method exhibits good robustness under varying numbers of encoder layers.

6. Conclusion

In this paper, we propose MemMambaAD, a memory-augmented state space model for multivariate time series anomaly detection. To address the problem of insufficient feature extraction of time series data, we design a sequence decomposition Mamba-TCN encoder to replace the traditional Transformer method for extracting temporal dependencies of multivariate time series, fully capturing the intrinsic patterns of multivariate time series with higher resolution. Additionally, to address the problem of insufficient storage of prototype patterns of normal data in the memory module, we designed a Dynamic Memory Update Mechanism, which combines memory items and memory increments adaptively through a Memory Selection Mechanism to record normal samples more accurately. Experimental results show that our method can improve the performance of time series anomaly detection, especially in complex data and changing environments. Compared with the baseline based on memory modules, MemMambaAD reduces the memory cost, which makes it have great potential for application in practical deployment, especially in resource-constrained environments.

Although MemMambaAD has demonstrated competitive performance across multiple evaluation metrics on various datasets, its accuracy has not yet reached the state-of-the-art level. Additionally, while MemMambaAD shows significant potential for deployment in resource-constrained environments, we have not yet conducted actual deployment experiments. In future work, we will explore more effective modeling techniques and prototype selection strategies, as well as evaluate the method's deployment performance on resource-constrained devices.

CRediT authorship contribution statement

Gang Li: Writing – original draft, Software, Funding acquisition, Formal analysis. **Mingchao Ge:** Writing – original draft, Supervision, Data curation, Conceptualization. **Jin Wan:** Writing – review & editing, Supervision, Investigation. **Delong Han:** Validation, Supervision. **Min Li:** Visualization, Investigation. **Mingle Zhou:** Writing – review & editing, Software, Project administration.

Funding

This work was supported by National Key Research and Development Program of China (2023YFB4004505), the Taishan Scholars Program (*NO*. tsqn202103097), the Taishan Industrial Experts Program (*NO*. tscy20221110), and the Qilu Youth Innovation Team (2024KJH028).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Abdulaal, A., Liu, Z., Lancewicki, T., 2021. Practical approach to asynchronous multivariate time series anomaly detection and localization. In: Data Min. Knowl. Discov.. pp. 2485–2494.

Ahamed, M.A., Cheng, Q., 2024. Timemachine: A time series is worth 4 mambas for long-term forecasting. arXiv preprint arXiv:2403.09898.

Bai, S., Kolter, J.Z., Koltun, V., 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271.

Deng, A., Hooi, B., 2021. Graph neural network-based anomaly detection in multivariate time series. In: AAAI. Vol. 35, (5), pp. 4027–4035.

Entezami, A., Sarmadi, H., Behkamal, B., Mariani, S., 2025. Early warning of structural damage via manifold learning-aided data clustering and non-parametric probabilistic anomaly detection. Mech. Syst. Signal. Process. 224, 111984.

Entezami, A., Shariatmadar, H., 2018. An unsupervised learning approach by novel damage indices in structural health monitoring for damage localization and quantification. Struct. Heal. Monit. 17 (2), 325–345.

Gong, D., Liu, L., Le, V., Saha, B., Mansour, M.R., Venkatesh, S., Hengel, A.v.d., 2019. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In: IEEE Conf. Comput. Vis. Pattern Recog.. pp. 1705–1714.

Gu, A., Dao, T., 2023. Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752.

Gu, A., Goel, K., Ré, C., 2021. Efficiently modeling long sequences with structured state spaces. arXiv preprint arXiv:2111.00396.

Guo, H., Zhou, Z., Zhao, D., Gaaloul, W., 2024. EGNN: Energy-efficient anomaly detection for IoT multivariate time series data using graph neural network. Future Gener. Comput. Syst. 151, 45–56.

Hundman, K., Constantinou, V., Laporte, C., Colwell, I., Soderstrom, T., 2018. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In: Data Min. Knowl. Discov.. pp. 387–395.

Kim, D., Park, S., Choo, J., 2024. When model meets new normals: Test-Time adaptation for unsupervised time-series anomaly detection. In: AAAI. Vol. 38, (12), pp. 13113–13121.

Kingma, D.P., Welling, M., 2013. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

Lai, K.-H., Zha, D., Xu, J., Zhao, Y., Wang, G., Hu, X., 2021. Revisiting time series outlier detection: Definitions and benchmarks. In: Adv. Neural Inform. Process. Syst..

Li, G., Jung, J.J., 2023. Deep learning for anomaly detection in multivariate time series: Approaches, applications, and challenges. Inf. Fusion 91, 93–102.

Li, Z., Zhao, Y., Han, J., Su, Y., Jiao, R., Wen, X., Pei, D., 2021. Multivariate time series anomaly detection and interpretation using hierarchical inter-metric and temporal embedding. In: Data Min. Knowl. Discov.. pp. 3220–3230.

Liang, A., Jiang, X., Sun, Y., Lu, C., 2024. Bi-Mamba4TS: Bidirectional mamba for time series forecasting. arXiv preprint arXiv:2404.15772.

Luo, D., Wang, X., 2024. Moderntcn: A modern pure convolution structure for general time series analysis. In: Int. Conf. Learn. Represent.

Lv, H., Chen, C., Cui, Z., Xu, C., Li, Y., Yang, J., 2021. Learning normal dynamics in videos with meta prototype network. In: IEEE Conf. Comput. Vis. Pattern Recog.. pp. 15425–15434.

Ma, J., Li, F., Wang, B., 2024. U-mamba: Enhancing long-range dependency for biomedical image segmentation. arXiv preprint arXiv:2401.04722.

Mathur, A.P., Tippenhauer, N.O., 2016. SWaT: A water treatment testbed for research and training on ICS security. In: 2016 International Workshop on Cyber-Physical Systems for Smart Water Networks (CySWater). IEEE, pp. 31–36.

Nakamura, T., Imamura, M., Mercer, R., Keogh, E., 2020. Merlin: Parameter-free discovery of arbitrary length anomalies in massive time series archives. In: 2020 IEEE International Conference on Data Mining. ICDM, IEEE, pp. 1190–1195.

Paparrizos, J., Boniol, P., Palpanas, T., Tsay, R.S., Elmore, A., Franklin, M.J., 2022. Volume under the surface: a new accuracy evaluation measure for time-series anomaly detection. PROC. VLDB ENDOW. 15 (11), 2774–2787.

Park, D., Hoshi, Y., Kemp, C.C., 2018. A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. IEEE Robot. Autom. Lett. 3 (3), 1544–1551.

Park, H., Noh, J., Ham, B., 2020. Learning memory-guided normality for anomaly detection. In: IEEE Conf. Comput. Vis. Pattern Recog.. pp. 14372–14381.

Qin, S., Luo, Y., Tao, G., 2023. Memory-augmented U-transformer for multivariate time series anomaly detection. In: ICASSP. IEEE, pp. 1–5.

Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S.A., Binder, A., Müller, E., Kloft, M., 2018. Deep one-class classification. In: Proc. Int. Conf. Mach. Learn.. PMLR, pp. 4393–4402.

Schölkopf, B., Williamson, R.C., Smola, A., Shawe-Taylor, J., Platt, J., 1999. Support vector method for novelty detection. Adv. Neural Inf. Process. Syst. 12.

Shen, L., Li, Z., Kwok, J., 2020. Timeseries anomaly detection using temporal hierarchical one-class network. Adv. Neural Inf. Process. Syst. 33, 13016–13026.

Song, J., Kim, K., Oh, J., Cho, S., 2024. Memto: Memory-guided transformer for multivariate time series anomaly detection. Adv. Neural Inf. Process. Syst. 36.

Su, Y., Zhao, Y., Niu, C., Liu, R., Sun, W., Pei, D., 2019. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In: Data Min. Knowl. Discov.. pp. 2828–2837.

Tuli, S., Casale, G., Jennings, N.R., 2022. Tranad: Deep transformer networks for anomaly detection in multivariate time series data. arXiv preprint arXiv:2201.07284.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. Adv. Neural Inf. Process. Syst. 30.

Wang, Z., Kong, F., Feng, S., Wang, M., Zhao, H., Wang, D., Zhang, Y., 2024. Is mamba effective for time series forecasting?. arXiv preprint arXiv:2403.11144.

Weston, J., Chopra, S., Bordes, A., 2014. Memory networks. arXiv preprint arXiv:1410.3916.

Wu, H., Xu, J., Wang, J., Long, M., 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. Adv. Neural Inf. Process. Syst. 34, 22419–22430.

Xu, J., Wu, H., Wang, J., Long, M., 2021. Anomaly transformer: Time series anomaly detection with association discrepancy. arXiv preprint arXiv:2110.02642.

Yang, Y., Li, Y., Zhang, T., Zhou, Y., Zhang, H., 2021. Early safety warnings for long-distance pipelines: A distributed optical fiber sensor machine learning approach. In: AAAI. Vol. 35, (17), pp. 14991–14999.

Yang, Y., Zhang, C., Zhou, T., Wen, Q., Sun, L., 2023. Dcdetector: Dual attention contrastive representation learning for time series anomaly detection. In: Data Min. Knowl. Discov.. pp. 3033–3045.

Yu, L.-r., Lu, Q.-h., Xue, Y., 2024. DTAAD: Dual TCN-attention networks for anomaly detection in multivariate time series data. Knowl.-Based Syst. 295, 111849.

Zhang, Y., Wang, J., Chen, Y., Yu, H., Qin, T., 2022. Adaptive memory networks with self-supervised learning for unsupervised anomaly detection. IEEE Robot. Autom. Lett..

Zhou, B., Liu, S., Hooi, B., Cheng, X., Ye, J., 2019. Beatgan: Anomalous rhythm detection using adversarially generated time series. In: IJCAI. Vol. 2019, pp. 4433–4439.

Zhu, L., Liao, B., Zhang, Q., Wang, X., Liu, W., Wang, X., 2024. Vision mamba: Efficient visual representation learning with bidirectional state space model. arXiv preprint arXiv:2401.09417.

Zong, B., Song, Q., Min, M.R., Cheng, W., Lumezanu, C., Cho, D., Chen, H., 2018. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In: Int. Conf. Learn. Represent.