STAR: Boosting Time Series Foundation Models for Anomaly Detection through State-Aware Adapter

Hanyin Cheng[1], **Ruitong Zhang**[1], **Yuning Lu**[1], **Yang Shu**[1], **Peng Chen**[2], **Yuan Jun**[2], **Meng Wang**[2], **Bin Yang**[1], **Chenjuan Guo**[1,✉]
[1]East China Normal University
[2]2012 APPLab, Huawei
{hycheng,rtzhang,ynlu}@stu.ecnu.edu.cn
{chenpeng192,yuanjun25,wangmeng71}@huawei.com
{yshu,byang,cjguo}@dase.ecnu.edu.cn

Abstract

Existing Time Series Foundation Models (TSFMs) for Multivariate Time Series Anomaly Detection often overlook discrete *state variables* that describe system status and treat them uniformly with *numerical variables*. This inappropriate modeling approach prevents the model from fully leveraging state information and even leads to significant performance degradation when state variables are integrated. Unlike numerical measurements, discrete states encode operating conditions that shape numerical behavior, motivating a dedicated representation and conditioning mechanism. To address this limitation, this paper proposes a novel **ST**ate-aware **A**dapte**R** (**STAR**). Specifically, STAR comprises three core innovative components: (1) an *Identity-guided State Encoder* that effectively captures the complex semantics of state variables; (2) a *Conditional Bottleneck Adapter* that dynamically injects state influence into TSFMs; and (3) a *Numeral-State Matching* module that effectively detects anomalies inherent to the state variables themselves. By separating state modeling from numerical processing, STAR adapts the model to system conditions while keeping the pretrained backbone frozen. Extensive experiments on five real-world datasets with four TSFM backbones show consistent improvements across diverse detection metrics, supporting the broad applicability of state-aware adaptation.

https://github.com/decisionintelligence/STAR

1 Introduction

In recent years, research into Time Series Foundation Models (TSFMs) has emerged as a significant area of interest [1–5]. Researchers are committed to building general-purpose models capable of understanding and processing a diverse range of time series tasks [6–8]. Benefiting from large-scale pre-training, TSFMs can deliver competitive performance without requiring extensive, task-specific fine-tuning. Multivariate Time Series Anomaly Detection (MTSAD), which focuses on identifying abnormal data in multivariate time series, is one of the key downstream tasks targeted by TSFMs [9–11]. It is applied widely in numerous critical domains, including financial fraud detection, medical disease identification, and cybersecurity threat detection [12–14].

Although TSFMs have achieved impressive performance in MTSAD, real-world industrial scenarios present more complex data manifestations. In these settings, variables exhibit more intricate forms. Beyond common *numerical variables* such as temperature, water flow, and velocity, *state variables* are also present and describe specific system states, such as the open/closed state of a valve [15], the

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

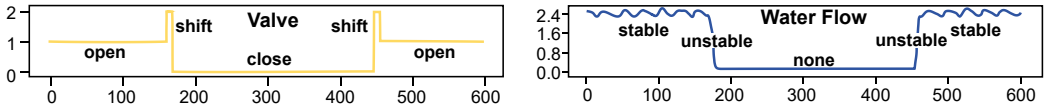


Figure 1: State variables (e.g., Valve) and numerical variables (e.g., Water Flow) in the SWaT.

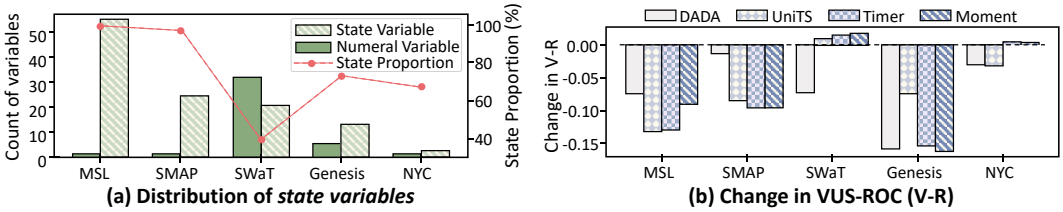


Figure 2: (a) State variables are prevalent in real-world datasets and frequently constitute a substantial proportion. (b) Impact of incorporating state variables on detection performance.

day of the week [16], or the positional state of a machine [17]. Unlike their numerical counterparts, these state variables are often organized as discrete data, where each value represents a specific category of that state. Moreover, they provide essential context about the system's operational state and directly influence the numerical variables' magnitude or temporal patterns. We refer to such influence as *condition-based influence*. For example, as shown in Figure 1, the valve's state (open, closed, or shifting) is a precondition for water flow and directly dictates the flow pattern (stable, unstable, or none). Figure 2a illustrates that state variables are widely prevalent in common datasets. However, due to either a lack of pre-training on state variables or uniform modeling for both numerical and state variables, **TSFMs often struggle to handle state variables**. As shown in Figure 2b, when state variables are incorporated into TSFMs through unified modeling, the detection performance fails to improve significantly or suffers a substantial decline.

To address this limitation, we aim to develop a plug-and-play method that enables TSFMs to leverage state variables better, thereby enhancing detection performance and facilitating adaptation to a wider range of industrial scenarios. However, this faces the following challenges:

First, how to extract meaningful embeddings from state variables? State variables possess complex semantics. For instance, within the SWaT [15], several state variables are primarily grouped into three clusters: switches of valves, pumps, and mercury lamps. Within a given group, state variables share similar semantics and exert similar physical influence on the system. Conversely, state variables across different groups are semantically heterogeneous, manifesting in disparate modes of influence. Furthermore, state variables also exhibit distinct temporal patterns, such as the stable periodic patterns of variables representing the day of the week [16]. In existing foundation models, state variables are typically embedded using linear layers pre-trained on numerical data. While this approach can partially capture the temporal features, it fails to adequately extract the rich and complex semantics inherent in their states.

Second, how to model the condition-based influence of state variables? Different from the correlation among variables [18, 19], the state variables act as preconditions that directly determine the magnitude or temporal patterns of the numerical variables. Therefore, for anomaly detection, it is essential to consider the condition-based influence when processing numerical variables. However, this asymmetric influence between state and numerical variables renders the modeling of condition-based influence a non-trivial task. Some TSFMs [8] model interactions among all variables in a unified manner. This approach erroneously treats them as equivalent, thereby failing to model the condition-based influence effectively.

To close these gaps, we propose **STAR**, a novel **ST**ate-aware **A**dapte**R** for TSFMs in the MTSAD to better model state variables during the fine-tuning phase. **First**, to better extract embeddings from state variables, we designed a *State Extractor* capable of simultaneously capturing both the state information and temporal patterns of state variables. This module incorporates an *Identity-guided State Encoder* which establishes a small-scale State Memory and retrieves a composite representation from the State Memory to encode state information guided by both Variable Identity and State Identity. This method effectively models the complex semantics of state variables. **Second**, we propose a

Conditional Bottleneck Adapter to effectively model the condition-based influence. The adapter dynamically generates adaptation parameters and bottleneck size based on state variables to modulate the TSFM's parameters, thereby directly influencing how the TSFMs process numerical variables. Building upon the aforementioned modules, we can capture meaningful state information and process numerical variables based on state. Furthermore, we have developed a *Numeral-State Matching* module to more effectively detect anomalies contained within the state variables.

Our contributions are summarized as follows:

- We design a universal State-aware Adapter that equips TSFMs with a more effective mechanism for handling state variables for the MTSAD.
- We propose a novel *Identity-guided State Encoder* in STAR to effectively model the complex homogeneous and heterogeneous semantics of state variables.
- We propose a novel *Conditional Bottleneck Adapter* in STAR, which can flexibly incorporate the condition-based influence of state variables in the TSFMs.
- We conducted extensive experiments on real-world datasets. The results show that STAR effectively improves the performance of TSFMs in MTSAD.

2 Related Work

2.1 Multivariate Time Series Anomaly Detection

Time series analysis has witnessed rapid progress across diverse tasks, including forecasting [20–29], anomaly detection [19, 30, 9], imputation [31, 32], and emerging reasoning-oriented tasks [33, 34]. Among them, multivariate time series anomaly detection (MTSAD) has received extensive attention across a variety of settings and modeling perspectives. Classic methods for MTSAD can be classified into non-learning [35–37], machine learning [38, 39], and deep learning [19, 40–43]. In particular, methods based on deep learning have seen significant success in recent years. They can be classified into forecasting-based [44], reconstruction-based [45–47, 43] and contrastive-based methods [48–50]. Although some of these models consider variable interactions within a unified modeling framework [19, 41, 51], or employ specialized modeling for discrete variables [52, 53], they still treat state variables as mere discrete numerical values, thereby overlooking the condition-based influence of state variables.

Moreover, the aforementioned methods are designed as isolated end-to-end architectures; they lack the inherent compatibility to be incorporated into TSFMs for capability augmentation.

2.2 TSFMs for Anomaly Detection

Researchers are actively exploring Foundation Models for various time series analysis tasks, including forecasting [11, 3, 54–56], classification [7], and anomaly detection [9].

For anomaly detection, models can be classified into two primary categories: 1) **Task-general models**: These models are typically pre-trained on extensive data, endowing them with excellent representation learning capabilities [8, 10, 6]. 2) **Task-specific models**: These models have architectures specifically designed for the anomaly detection task. For instance, DADA [9] utilizes multiple bottleneck layers of varying dimensions to enhance adaptability to anomalous data. However, these TSFMs lack explicit state variable modeling, resulting in constrained performance in complex real-world applications.

Additionally, some work has focused on designing adapters to augment TSFMs during the fine-tuning phase, such as AdaPTS [57] and MSFT [58]. However, these efforts are primarily concentrated on forecasting tasks, and more importantly, fail to enable TSFMs to better process state variables.

3 STAR

For the task of time series anomaly detection, we consider a time series as $\boldsymbol{X} = \{\boldsymbol{X}^n, \boldsymbol{X}^s\} \in \mathbb{R}^{T\times(C_n+C_s)}$, which consists of C_n numerical variables $\boldsymbol{X}^n$ and C_s state variables $\boldsymbol{X}^s$ over T time points. A TSFM is leveraged to identify anomalies in a previously unseen test series $\boldsymbol{X}_{\text{test}}$ either via direct inference or subsequent fine-tuning. Specifically, the model's goal is to generate a binary

prediction sequence $\boldsymbol{Y}_{\text{test}} = (y_1, y_2, \ldots, y_{T_{\text{test}}}) \in \mathbb{R}^{T_{\text{test}}}$. Each element $y_t \in \{0, 1\}$ in this sequence serves as a label, signifying whether the observation at time t is anomalous.

3.1 Overview

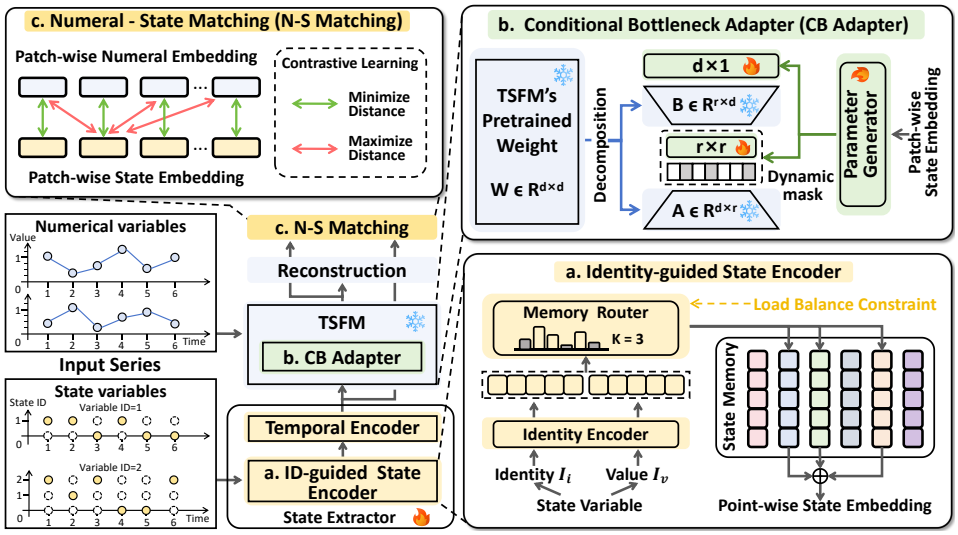


Figure 3: The overview of STAR.

Figure 3 shows the overall architecture of **STAR**, which reconsiders the state variables for TSFMs using the **ST**ate-aware **A**dapte**R**. We shift the processing of state variables from TSFMs to STAR. It mainly consists of three modules: 1) **State Extractor**. Addressing the specific characteristics of state variables, this module simultaneously extracts their state and temporal features and utilizes an *Identity-guided Selective State Encoder* to model the complex semantics of state variables effectively. 2) **Conditional Bottleneck Adapter**. To better model the condition-based influence, we no longer directly model interactions between state and numerical variables. Instead, we use the state variables to modulate the parameter values and bottleneck sizes during training, thereby influencing its processing of numerical variables. 3) **Numeral-State Matching**. To better detect anomalies in the state, we design a contrastive learning objective based on the degree of matching between numerical and state variables at the patch level. Notably, during the fine-tuning process, the entire Backbone network is frozen, and only STAR is utilized to perform the fine-tuning effectively.

3.2 State Extractor Guided by Identity

The *State Extractor* operates in two sequential stages. First, an *Identity-guided State Encoder* processes each state variable to capture its information from both the state identity (i.e., index of the state variable) and the state value (i.e., discrete value of the state variable) at each time point. Subsequently, a *Temporal Encoder* extracts temporal patterns within each patch.

3.2.1 Identity-guided State Encoder

Formally, state variables are a type of categorical variable, and different state variables can have different cardinalities, i.e., different numbers of categories. Most existing methods learn a representation for every individual category via a dedicated, learnable vector to embed the categorical variable [59–61]. This method, however, imposes a substantial parameter burden and learning complexity. More importantly, it fails to account for the similarity or heterogeneity among different state variables.

To address the complex semantics of state variables associated with grouping patterns, we propose a novel router network that dynamically assigns different state variables that have distinct state values into clusters. Our approach introduces a learnable, compact State Memory, denoted as $\boldsymbol{S} \in \mathbb{R}^{N \times d}$, which consists of N d-dimensional vectors corresponding to distinct groups. Under the guidance of the embedding of state identity $\boldsymbol{I}_i \in \mathbb{R}^{T \times C_s \times d}$ and the embedding of state value $\boldsymbol{I}_v \in \mathbb{R}^{T \times C_s \times d}$, we select a vector subset from the State Memory and utilize the linear combination of its vectors to

represent the state variable. We begin by employing a Sinusoidal Encoding [62] to transform the state identity and state value into $\boldsymbol{I}_i$ and $\boldsymbol{I}_v$:

$$\boldsymbol{I}_i[:,i,k] = \begin{cases} \sin\left(\frac{i}{\lambda^{2n/d}}\right), & \text{if } k = 2n \\ \cos\left(\frac{i}{\lambda^{2n/d}}\right), & \text{if } k = 2n+1 \end{cases}, \ \boldsymbol{I}_v[:,i,k] = \begin{cases} \sin\left(\frac{\boldsymbol{X}^S[:,i]}{\lambda^{2n/d}}\right), & \text{if } k = 2n \\ \cos\left(\frac{\boldsymbol{X}^S[:,i]}{\lambda^{2n/d}}\right), & \text{if } k = 2n+1 \end{cases}, \quad (1)$$

where λ is a constant; i and k represent the index of the variable and the hidden dimension of the embedding, respectively; $n \in \{0, 1, 2, 3 \dots\}$ represents any natural number with $2n$ and $2n+1$ corresponding to even and odd components; and $\boldsymbol{X}^S[:,i]$ denotes the discrete value of the i-th state variable at the corresponding time point. The *Memory Router* leverages $\boldsymbol{I}_i$ and $\boldsymbol{I}_v$ to dynamically choose a subset of K vectors from the State Memory. To preserve a continuous gradient flow, we implement the selection mechanism using a differentiable Top-K routing:

$$\boldsymbol{W}_s = f(\text{concat}(\boldsymbol{I}_i, \boldsymbol{I}_v)) \in \mathbb{R}^{T \times C_s \times N}, \ \boldsymbol{\theta} = \text{topK}(\boldsymbol{W}_s) \in \mathbb{R}^{T \times C_s \times 1}, \quad (2)$$

$$\boldsymbol{W}_{\text{inf}} = \log(\text{sigmoid}((\boldsymbol{W}_s - \boldsymbol{\theta})/\epsilon)), \ \boldsymbol{W}_{\text{route}} = \text{softmax}(\boldsymbol{W}_s + \boldsymbol{W}_{\text{inf}}) \in \mathbb{R}^{T \times C_s \times N}, \quad (3)$$

where f, implemented as an MLP, learns the selection weights from the concatenated embeddings. The function topK returns the K-th largest value, denoted as θ. ϵ is a small positive constant that pushes the sigmoid's output to its extremes (0 or 1). $\boldsymbol{W}_{\text{route}}$ represents the route weights for memory slots. $\boldsymbol{W}_{\text{inf}}$ is a mask matrix, where all corresponding values in $\boldsymbol{W}_s$ smaller than $\boldsymbol{\theta}$ are attenuated towards zero. The point-wise state embedding $\boldsymbol{S}_{\text{point}} \in \mathbb{R}^{T \times C_s \times d}$ is obtained by: $\boldsymbol{S}_{\text{point}} = \boldsymbol{W}_{\text{route}} \otimes \boldsymbol{S}$.

To further promote the comprehensive training of all elements within the State Memory and to prevent routing imbalance, we introduce a constraint based on the squared coefficient of variation (CV^2) [63]. This constraint is applied to both the selection frequency $\boldsymbol{E}_{\text{sel}}$ and the routing importance of each element $\boldsymbol{E}_{\text{imp}}$ to ensure load balancing:

$$\boldsymbol{E}_{\text{sel}} = \text{avg}_{T,C_s}(\text{sigmoid}((\boldsymbol{W}_s - \boldsymbol{\theta})/\epsilon)) \in \mathbb{R}^N, \ \boldsymbol{E}_{\text{imp}} = \text{avg}_{T,C_s}(\boldsymbol{W}_{\text{route}}) \in \mathbb{R}^N, \quad (4)$$

$$\mathcal{L}_{\text{bal}} = \frac{\text{var}_N(\boldsymbol{E}_{\text{sel}})}{(\text{avg}_N(\boldsymbol{E}_{\text{sel}}))^2} + \frac{\text{var}_N(\boldsymbol{E}_{\text{imp}})}{(\text{avg}_N(\boldsymbol{E}_{\text{imp}}))^2}, \quad (5)$$

where $\text{avg}_{T,C_s}(\cdot)$ denotes average aggregation along the temporal and variable dimensions, $\text{avg}_N(\cdot)$ and $\text{var}_N(\cdot)$ are functions that compute the mean and variance of $\boldsymbol{E}_{\text{sel}}$ and $\boldsymbol{E}_{\text{imp}}$, respectively. $\mathcal{L}_{\text{bal}}$ denotes the load-balancing loss, which is incorporated into the overall training objective.

3.2.2 Temporal Encoder

Motivated by the observation that state variables often exhibit temporal patterns, such as the strong periodicity representing the day of the week, we further model the temporal interactions among the point-wise state embedding $\boldsymbol{S}_{\text{point}}$ to derive patch-wise state embeddings $\boldsymbol{S}_{\text{patch}}$. Initially, we segment $\boldsymbol{S}_{\text{point}}$ into m patches of length l in a manner consistent with the backbone's configuration, yielding $\boldsymbol{P} \in \mathbb{R}^{m \times l \times C_s \times d}$. This process comprises two components: intra-patch and inter-patch interaction:

$$\boldsymbol{P}_{\text{intra}} = f_1(\boldsymbol{P}) \in \mathbb{R}^{m \times C_s \times d}, \ \boldsymbol{P}_{\text{inter}} = f_2(\boldsymbol{P}_{\text{intra}}) + \boldsymbol{P}_{\text{intra}} \in \mathbb{R}^{m \times C_s \times d}, \quad (6)$$

where $f_1 : \mathbb{R}^{l \times d} \to \mathbb{R}^d$ aggregates the representations within each patch, and $f_2 : \mathbb{R}^m \to \mathbb{R}^m$ extracts valuable contextual information for each patch. Each state variable reflects only a partial aspect of the system's information. Therefore, to construct a more holistic representation of the system's overall state, we further perform an aggregation over these variables inside each patch:

$$\boldsymbol{W}_{\text{agg}} = f_3(\boldsymbol{P}_{\text{inter}}) \in \mathbb{R}^{m \times C_s}, \ \boldsymbol{S}_{\text{patch}} = f_4(\text{sum}_{C_s}(\boldsymbol{W}_{\text{agg}} \odot \boldsymbol{P}_{\text{inter}})) \in \mathbb{R}^{m \times d}, \quad (7)$$

where $f_3 : \mathbb{R}^d \to \mathbb{R}^1$ is a function that adaptively computes the aggregation weights, and $f_4 : \mathbb{R}^d \to \mathbb{R}^d$ serves to refine the resulting fused representation further; $\odot$ denotes element-wise multiplication. For a lightweight implementation, all four functions, f_1, f_2, f_3, and f_4, are implemented as MLPs.

3.3 Conditional Bottleneck Adapter

To efficiently fine-tune pre-trained parameters, the weight update $\Delta\boldsymbol{W}$ for a pre-trained matrix $\boldsymbol{W}_0 \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ is typically constrained to a low-rank decomposition [64]. This process can be formalized as follows:

$$\boldsymbol{h} = \boldsymbol{W}_0\boldsymbol{x} + \Delta\boldsymbol{W}\boldsymbol{x}, \Delta\boldsymbol{W} = \boldsymbol{A}\boldsymbol{B}, \quad (8)$$

where $\boldsymbol{h}$ and $\boldsymbol{x}$ denote the output and input of the module, respectively. $\boldsymbol{A} \in \mathbb{R}^{d_{\text{in}} \times r}$ and $\boldsymbol{B} \in \mathbb{R}^{r \times d_{\text{out}}}$ are two trainable low-rank matrices, with the rank $r \ll \min(d_{\text{in}}, d_{\text{out}})$.

Leveraging the efficiency of low-rank decomposition, we innovatively propose the Conditional Bottleneck Adapter. This module perceives the system's state to dynamically generate its parameters and determine its bottleneck size, as illustrated in Figure 3b. The weight update $\Delta \boldsymbol{W}$ is given by the following equation:

$$\Delta \boldsymbol{W} = \boldsymbol{ARB} \odot \boldsymbol{D}, \;\; \boldsymbol{R} \in \mathbb{R}^{r \times r}, \boldsymbol{D} \in \mathbb{R}^{d_{\text{out}} \times 1}, \tag{9}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are decomposed from $\boldsymbol{W}_0$, as detailed in Section 3.3.1, and remain frozen. $\boldsymbol{R}, \boldsymbol{D}$ are determined by the state variable as detailed in Section 3.3.2.

Different from selecting the bottleneck size with the Mixture of Experts architecture [9], which requires more parameters, we propose a novel dynamic mask on the $\boldsymbol{R}$ matrix to modulate the bottleneck size efficiently. This mechanism enables the model to adaptively adjust the bottleneck size of training parameters conditioned on the state variables, thereby flexibly controlling model capacity to alleviate overfitting and underfitting in anomaly detection [65].

With the low-rank matrix decomposition and the learnable matrix $\boldsymbol{R}$ conditioned on the system's state, our method avoids additional performance overhead. Theoretically, our adapter can be integrated with the linear layers of any module.

3.3.1 Pretrained Weight Decomposition

To preserve the performance of the original parameters as much as possible, we employ Singular Value Decomposition (SVD) [66] to decompose the pretrained weight:

$$\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V} = \text{SVD}(\boldsymbol{W}_0), \boldsymbol{U} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}}, \; \boldsymbol{\Sigma} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}, \boldsymbol{V} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{out}}}, \tag{10}$$

where $\boldsymbol{\Sigma}$ is the singular value matrix, which we truncate to its leading r values to form the matrix $\boldsymbol{\Sigma}_1 \in \mathbb{R}^{d_{\text{in}} \times r}$ and $\boldsymbol{\Sigma}_2 \in \mathbb{R}^{r \times d_{\text{out}}}$ for the purpose of low-rank approximation. Then we compute the low-rank matrices as :

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}_1 \in \mathbb{R}^{d_{\text{in}} \times r}, \boldsymbol{B} = \boldsymbol{\Sigma}_2 \boldsymbol{V} \in \mathbb{R}^{r \times d_{\text{out}}}. \tag{11}$$

3.3.2 Parameter Generator and Adaptive Bottleneck Size

For the t-th patch, we utilize its corresponding $\boldsymbol{S}^t_{\text{patch}}$ to compute the state-dependent parameters $\boldsymbol{R}$ and $\boldsymbol{D}$. It is worth noting that the $\boldsymbol{R}$ and $\boldsymbol{D}$ matrices are patch-specific, as they are dynamically generated based on the distinct state of each individual patch:

$$\boldsymbol{R}_{\text{init}}, \boldsymbol{D} = g_1(\boldsymbol{S}^t_{\text{patch}}), g_1 := \mathbb{R}^d \to \mathbb{R}^{(r \times r + d_{\text{out}})}. \tag{12}$$

As shown in Figure 3b, we adaptively learn a state-aware dynamic mask matrix based on state variables via a soft-masking mechanism. We apply this mask to the vector $\boldsymbol{R}_{\text{init}}$ and then insert the masked vector between the low-rank matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ to modulate the bottleneck size of the Conditional Bottleneck (CB) Adapter. This process can be formalized as follows:

$$\boldsymbol{R}_{\text{radio}} = g_2(\boldsymbol{S}^t_{\text{patch}}) \in \mathbb{R}^r, \Gamma = g_3(\boldsymbol{S}^t_{\text{patch}}), \tag{13}$$

$$\boldsymbol{R} = \text{sigmoid}((\boldsymbol{R}_{\text{radio}} - \Gamma)/\epsilon) \odot \boldsymbol{R}_{\text{init}} \in \mathbb{R}^r, \tag{14}$$

where $g_2 := \mathbb{R}^d \to \mathbb{R}^r$, $g_3 := \mathbb{R}^d \to \mathbb{R}^1$. g_2, g_3 are implemented as MLPs. $\boldsymbol{R}_{\text{radio}}$ is score for each row in $\boldsymbol{R}$, Γ is a masking threshold. Leveraging a soft mask mechanism (Eq 14), the $\boldsymbol{R}$ matrix adaptively controls the bottleneck size according to the system state, allowing for the dynamic handling of numerical variables.

3.4 Pipeline with Numeral-State Matching

Anomalies may manifest not only as aberrant temporal patterns in numerical variables but also as abnormal conditions within the state variables. Existing TSFMs, however, detect anomalies solely through time series reconstruction. Since state variables are represented as categorical variables, merely reconstructing their state identities is insufficient for accurately identifying state-based anomalies. To address this limitation, we introduce the *Numeral-State Matching* module. This module is designed to detect anomalies by learning the degree of correspondence between numerical variables and their associated state variables.

3.4.1 Finetuning phase

Constructing positive and negative pairs for contrastive learning by injecting anomalies introduces significant training overhead and makes it difficult to guarantee the effectiveness of the synthetic anomalies [40, 67]. To circumvent these issues, we adopt a different approach. As shown in Figure 3c, we define a positive pair as the corresponding patch-wise numeral embedding $\boldsymbol{N}^t_{\text{patch}} \in \mathbb{R}^d$ and patch-wise state embedding $\boldsymbol{S}^t_{\text{patch}} \in \mathbb{R}^d$. All other non-corresponding pairings are consequently treated as negative examples. Instead of contrasting among different views of the same variable [68, 69], our method focuses on learning the correlations between state and numerical variables, and subsequently uses this as a basis for detecting potential anomalies. The Numeral-State Matching contrastive loss can be expressed as follows:

$$\mathcal{L}_{\text{match}} = -\frac{1}{M}\sum_{t=1}^{M}\log\left(\frac{\mathcal{G}(\boldsymbol{N}^t_{\text{patch}}, \boldsymbol{S}^t_{\text{patch}})}{\sum_{k=1}^{M}\mathcal{G}(\boldsymbol{N}^k_{\text{patch}}, \boldsymbol{S}^t_{\text{patch}})}\right), \tag{15}$$

where M denotes the total number of patches in one batch; $\boldsymbol{N}^i_{\text{patch}}$ and $\boldsymbol{S}^i_{\text{patch}}$ are the respective numeral and state embeddings corresponding to the i-th patch; $\mathcal{G} := \exp(\text{sim}(\cdot)/\tau)$; the function $\text{sim}(\cdot)$ is the cosine similarity, and τ is a temperature coefficient used to scale the logits in the contrastive loss.

The overall loss during training is composed of three components: the numerical variable reconstruction loss ($\mathcal{L}_{\text{rec}}$), the load-balancing loss ($\mathcal{L}_{\text{bal}}$), and the Numeral-State Matching loss ($\mathcal{L}_{\text{match}}$). The total loss is formulated as: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{rec}} + \lambda_1\mathcal{L}_{\text{bal}} + \lambda_2\mathcal{L}_{\text{match}}$, where λ_1 and λ_2 are parameters that balance the contribution of each loss term.

3.4.2 Inference phase

Anomaly detection is performed in two parts: point-wise *numerical reconstruction* and patch-wise *Numeral-State Matching*. During inference on a test time series, their respective scores, denoted as $\text{score}_{\text{rec}}$ and $\text{score}_{\text{match}}$, are calculated separately:

$$\text{score}_{rec} = \text{avg}_{C_n}(\text{mse}(\hat{\boldsymbol{X}}^n_{\text{test}}, \boldsymbol{X}^n_{\text{test}})) \in \mathbb{R}^{T_{\text{test}}},\ \text{score}_{match} = 1 - \text{sim}(\boldsymbol{N}_{\text{patch}}, \boldsymbol{S}_{\text{patch}}) \in \mathbb{R}^{m_{\text{test}}}, \tag{16}$$

where $\text{mse}(\cdot)$ is the mean squared error, $\text{sim}(\cdot)$ denotes the cosine similarity, $\hat{\boldsymbol{X}}^n$ represents the reconstruction output of numerical variables from the backbone, and T_{test} and m_{test} are the total number of time points and patches in the test set, respectively. $\text{avg}_{C_n}(\cdot)$ is an operation that computes the mean value across the variable dimension.

Given that $\text{score}_{\text{rec}}$ and $\text{score}_{\text{match}}$ differ in both temporal granularity and numerical scale, we first apply linear interpolation ($LI(\cdot)$) to $\text{score}_{\text{match}}$ to align their resolutions. Subsequently, the two scores are fused by taking their element-wise product:

$$\text{score}'_{\text{match}} = \text{LI}(\text{score}_{\text{match}}) \in \mathbb{R}^{T_{\text{test}}},\ \text{score}_{\text{total}} = \text{score}_{rec} \times \text{softmax}(\text{score}'_{\text{match}}) \in \mathbb{R}^{T_{\text{test}}}. \tag{17}$$

Our approach not only detects anomalies in numerical variables but also detects anomalies in the system state. Furthermore, by leveraging patch-wise similarity computation, the model's capability to detect subsequence anomalies is effectively enhanced.

4 Experiments

4.1 Experimental Settings

Datasets. To evaluate our method, we selected five classic real-world multivariate anomaly detection datasets: 1) **MSL** [70]: Spacecraft incident and anomaly data from the MSL Curiosity rover. 2) **SMAP** [70]: Contains telemetry data from NASA's Soil Moisture Active Passive (SMAP) satellite. 3) **SWaT** [15]: Obtained from 51 sensors of the critical infrastructure system under continuous operations. 4) **Genesis** [17]: A sensor and control signals dataset collected from cyber-physical production systems. 5) **NYC** [16]: The Transportation dataset provides information on taxi and ride-hailing trips in New York. Each of these datasets contains a significant number of *state variables*, with

Table 1: Statistics of multivariate datasets (AR: anomaly ratio).

Dataset	Domain	Numeral Numbers	State Numbers	AR (%)	Avg Total Length	Avg Test Length
MSL	Spacecraft	1	54	5.88	132,046	73,729
SMAP	Spacecraft	1	24	9.72	562,800	427,617
SWaT	Water treatment	28	23	5.78	944,919	449,919
Genesis	Machinery	5	13	0.31	16,220	12,616
NYC	Transport	1	2	0.57	17,520	4,416

detailed statistics presented in Table 1. More details about the datasets are available in Section A.1 of Appendix A.

Backbones. In the realm of time series analysis, numerous models have surfaced in recent years. We choose models designed for multiple tasks or specialized for anomaly detection, including task-specific model DADA [9] and task-general models: UniTS [8], Moment [10], Timer [6]. The detailed description is available in Section A.2 of Appendix A.

Metrics. To circumvent the potential for unfair comparisons arising from the diverse threshold selection methods employed by different TSFMs, we primarily adopt threshold-irrelevant metrics for evaluation. These include the AUC-ROC (A-R), VUS-ROC (V-R), and VUS-PR (V-P) [71]. Meanwhile, the comprehensive evaluation is also supplemented by a range of commonly used metrics mentioned in TAB[72]. The description and implementation details of all metrics are shown in Section A.3 of Appendix A.

4.2 Detection results

Main results. We evaluate STAR on 5 real-world datasets containing state variables, using 4 TSFMs as backbones. We uniformly use 10% of the downstream data for fine-tuning. The results are summarized in Table 2. For a fair comparison, we evaluated two baseline fine-tuning methods: (1) **Backbone**: standard fine-tuning, and (2) **LoRA**: fine-tuning the base model with LoRA [64]. It can be seen that LoRA-based fine-tuning offers inconsistent and often marginal performance changes. Our method consistently delivers substantial enhancements to the detection performance of TSFMs. Compared to standard fine-tuning, STAR improves the V-R metric on task-specific and task-general TSFMs by 9.13% and 4.19%, respectively. This demonstrates the generalizability of STAR. The validity of our motivation and modeling is further highlighted on Genesis and SWaT, where STAR delivers larger improvements of 7.42% and 14.34%, owing to the meaningful physical interpretations of their state variables. Notably, despite the significant performance discrepancies on the SWaT dataset, this variance is considered to be within a reasonable range due to the unique characteristics of its distribution. A similar phenomenon can be found in [19].

Table 2: Results of adapters for different TSFMs. For each backbone, the best result for each metric is highlighted in **bold**.

Dataset	MSL			SMAP			SWaT			Genesis			NYC		
Metric	A-R	V-R	V-P	A-R	V-R	V-P	A-R	V-R	V-P	A-R	V-R	V-P	A-R	V-R	V-P
DADA	0.697	0.753	0.305	0.421	0.464	0.129	0.811	0.684	0.513	0.839	0.861	0.157	0.555	0.678	0.077
+LoRA	0.714	0.757	0.306	0.435	0.475	0.134	0.799	0.675	0.516	0.819	0.846	0.155	0.579	0.687	0.077
+STAR	**0.725**	**0.768**	**0.308**	**0.491**	**0.529**	**0.141**	**0.836**	**0.824**	**0.634**	**0.930**	**0.938**	**0.202**	**0.598**	**0.695**	**0.079**
UniTS	0.745	0.794	0.294	0.532	0.571	0.149	0.237	0.339	0.127	0.696	0.763	0.016	0.542	0.655	0.055
+LoRA	0.742	0.792	0.296	0.530	0.573	0.151	0.235	0.335	0.125	0.693	0.767	0.017	0.561	0.660	0.057
+STAR	**0.755**	**0.802**	**0.318**	**0.536**	**0.576**	**0.153**	**0.273**	**0.372**	**0.139**	**0.788**	**0.857**	**0.060**	**0.569**	**0.710**	**0.062**
Moment	0.739	0.786	0.292	0.525	0.564	0.148	0.229	0.330	0.121	0.725	0.794	0.141	0.592	0.726	**0.156**
+LoRA	0.741	0.788	0.297	0.524	0.564	0.147	0.232	0.336	0.124	0.727	0.792	0.142	0.595	0.730	0.152
+STAR	**0.744**	**0.794**	**0.317**	**0.534**	**0.572**	**0.156**	**0.267**	**0.367**	**0.130**	**0.790**	**0.843**	**0.144**	**0.632**	**0.746**	0.121
Timer	0.732	0.772	0.309	0.552	0.596	0.161	0.232	0.335	0.133	0.953	0.938	0.193	**0.680**	0.749	0.070
+LoRA	0.729	0.763	0.307	0.550	0.598	0.157	0.238	0.346	0.130	0.964	0.947	0.204	0.593	0.714	0.063
+STAR	**0.748**	**0.781**	**0.312**	**0.556**	**0.603**	**0.165**	**0.265**	**0.367**	**0.136**	**0.974**	**0.967**	**0.267**	0.665	**0.762**	**0.076**

Multi-metrics results. To further assess detection performance, we evaluate STAR using range-based metrics [73], affiliation-based metrics [74], and threshold-independent metrics [71] on MSL and

Genesis. Table 3 shows improvements in 50 of the 52 comparisons, including 12 of 13 metrics for DADA on each dataset. On MSL, DADA's range-based F1 increases from 0.220 to 0.226 and affiliation-based F1 from 0.782 to 0.791; on Genesis, UniTS's range-based F1 rises from 0.053 to 0.074. Genesis AUC-ROC also increases from 0.839 to 0.930 for DADA and from 0.696 to 0.788 for UniTS, confirming gains in both detection and anomaly score ranking.

Table 3: Multi-metrics results on real-world datasets. For each dataset and backbone, the better result is highlighted in **bold**; ties are also highlighted.

Dataset	Method	Threshold-relevant							Threshold-irrelevant					
		ACC	R-P	R-R	R-F1	Aff-P	Aff-R	Aff-F1	A-R	A-P	R-A-R	R-A-P	V-R	V-P
MSL	DADA	0.813	0.199	0.245	0.220	0.661	0.957	0.782	0.697	0.244	0.759	**0.309**	0.753	0.305
	+STAR	**0.843**	**0.208**	**0.248**	**0.226**	**0.670**	**0.965**	**0.791**	**0.725**	**0.246**	**0.776**	0.305	**0.768**	**0.308**
	UniTS	0.813	0.219	0.231	0.225	0.668	0.960	0.788	0.745	0.218	0.802	0.301	0.794	0.294
	+STAR	**0.816**	**0.222**	**0.236**	**0.228**	**0.669**	**0.961**	**0.789**	**0.755**	**0.238**	**0.810**	**0.326**	**0.802**	**0.318**
Genesis	DADA	0.991	0.061	**0.187**	0.092	0.782	0.977	0.869	0.839	0.180	0.873	0.155	0.861	0.157
	+STAR	**0.993**	**0.173**	0.174	**0.174**	**0.828**	**0.998**	**0.905**	**0.930**	**0.235**	**0.948**	**0.204**	**0.938**	**0.202**
	UniTS	0.985	0.032	0.156	0.053	0.706	0.951	0.811	0.696	0.016	0.784	0.017	0.763	0.016
	+STAR	**0.991**	**0.047**	**0.173**	**0.074**	**0.786**	**0.971**	**0.869**	**0.788**	**0.042**	**0.874**	**0.061**	**0.857**	**0.060**

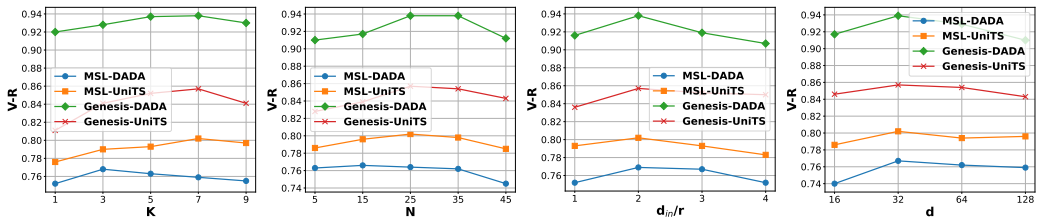
Table 4: Ablation studies for STAR. The better results are highlighted in **bold**.

	Dataset			MSL				Genesis			
	ID-guided State Encoder	CB Adapter	N - S Matching	DADA		UniTS		DADA		UniTS	
				V-R	V-P	V-R	V-P	V-R	V-P	V-R	V-P
1	✗	✗	✗	0.753	0.305	0.794	0.294	0.861	0.157	0.763	0.016
2	✗	✓	✗	0.757	0.303	0.793	0.298	0.876	0.160	0.786	0.023
3	✗	✗	✓	0.761	0.296	0.792	0.304	0.873	0.162	0.784	0.018
4	✓	✓	✗	0.764	0.307	0.800	0.312	0.912	0.184	0.827	0.022
5	✓	✗	✓	0.765	0.304	0.797	0.307	0.897	0.176	0.812	0.038
6	✓	✓	✓	**0.768**	**0.308**	**0.802**	**0.318**	**0.938**	**0.202**	**0.857**	**0.060**

4.3 Model Analysis

Ablation Study. To analyze the contribution of each module, we conducted an ablation study, detailed in Table 4. Here, ✗ and ✓ indicate a module removed and added, respectively. Row 1 shows the standard fine-tuning, while Rows 2-6 illustrate the performance when integrating specific modules of STAR with the backbone. We have the following observations: (1) A comparison of rows 1-3 reveals that both the *Conditional Bottleneck Adapter* (*CB Adapter*) and the *Numeral-State Matching* (*N-S Matching*) module can enhance the backbone's performance to a certain degree. This limited improvement can likely be attributed to the lack of meaningful state information. (2) In Rows 4 and 5, the addition of the *ID-guided State Encoder* results in a more pronounced performance enhancement with the two modules. This validates the effectiveness of the *ID-guided State Encoder* in capturing complex state information. (3) The combination of all three modules creates a virtuous cycle, resulting in the best performance, as shown in Row 6. Specifically, the *N-S Matching* module optimizes the state representations, which in turn enhances the effectiveness of the other modules.

Parameter Sensitivity. We study the parameter sensitivity of STAR, including the number of selected vectors in *Memory Router* (K), the total number of memory vectors (N), the rank of the adapter (r), and the hidden dimension size of the adapter (d). To enhance the flexibility of STAR, we search for the value of d_{in}/r directly, rather than searching for r itself. Figure 4a shows the impact of K. In most cases, a smaller K is insufficient for effective learning. We typically select 7 as the preferred option. Besides, Figure 4b illustrates the impact of N. Common choices for this value are 15 or 25. This approach is more lightweight compared to defining a learnable vector for each state of each state variable [59]. As shown in Figure 4c, by reducing the d_{in} to $d_{in}/2$ or $d_{in}/3$, we can achieve a balance between efficiency and effectiveness. Figure 4d indicates that a large hidden dimension is not required, thus avoiding substantial additional computational costs.



(a) Memory vectors (b) Number of memory slots (c) Rank of adapter (d) Hidden dimension size

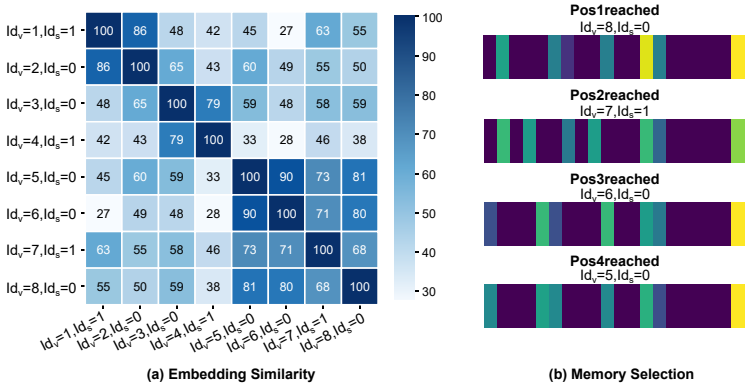
Figure 4: Parameter sensitivity studies of main hyper-parameters in STAR.

Visualizations. To further validate the effectiveness of the *Identity-guided State Encoder*, we visualized its output embeddings as well as the selections made by the *Memory Router*. We selected Genesis [17] for our analysis, and a subset of its variables is described in Table 5.

We use $\mathrm{Id_v}$ to denote the variable identity (index of *state variable*), and $\mathrm{Id_s}$ to denote the state identity (discrete value of *state variable*). Figure 5a illustrates the similarity between embeddings of different variables, where a distinct clustering pattern is observed. This indicates that variables with similar semantic information exhibit higher similarity. Since our router is conditioned on both variable identity and state identity, embeddings for variables that share the same variable identity or state identity exhibit greater similarity. However, the overall influence of variable identity is more pronounced than that of state identity. These observations are further corroborated by Figure 5b.

Table 5: Descriptions of selected state variables in Genesis.

$\mathrm{Id_v}$	Group	Variable name	Descriptions
1	Slider State	Slider_OUT	The slider is on the outside.
2		Slider_IN	The slider is on the inside.
3	Detection State	NonMetall	Non-metal detected.
4		Metall	Metal detected.
5	Position State	Pos4reached	Arrived at Position 4.
6		Pos3reached	Arrived at Position 3.
7		Pos2reached	Arrived at Position 2.
8		Pos1reached	Arrived at Position 1.



(a) Embedding Similarity (b) Memory Selection

Figure 5: (a) shows the similarity between the embeddings output from *Identity-guided State Encoder*. (b) shows the weight of selection in *Memory Router*.

More Analysis. To demonstrate the applicability of STAR, we also evaluate its performance on datasets containing only numerical variables (B.1). Moreover, we provide additional visualizations of bottleneck behavior across different states in Appendix B.3. Finally, we present a case study to highlight the importance of state variables for anomaly detection in Appendix B.2.

5 Conclusion

In this study, we propose STAR to address the failure of TSFMs to handle state variables. We propose the Identity-guided State Encoder and the Conditional Bottleneck Adapter to capture the complex semantics of states and the conditional influence of state variables, respectively. Our implementation is available at https://github.com/decisionintelligence/STAR.

Acknowledgments and Disclosure of Funding

This work was partially supported by National Natural Science Foundation of China (62372179). Chenjuan Guo is the corresponding author of the work.

References

[1] Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-moe: Billion-scale time series foundation models with mixture of experts. *arXiv e-prints*, pages arXiv–2409, 2024.

[2] Hanyin Cheng, Xingjian Wu, Yang Shu, Zhongwen Rao, Lujia Pan, Bin Yang, and Chenjuan Guo. CoRA: Boosting time series foundation models for multivariate forecasting through correlation-aware adapter. In *ICLR*, 2026.

[3] Yong Liu, Guo Qin, Zhiyuan Shi, Zhi Chen, Caiyin Yang, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Sundial: A family of highly capable time series foundation models. *arXiv preprint arXiv:2502.00816*, 2025.

[4] Hanyin Cheng, Linfeng Wang, Zhengbo Qu, Yang Shu, Zhongwen Rao, Meng Wang, Yijie Li, Xin Jiang, Bin Yang, and Chenjuan Guo. QiYao-M: Multimodal time series foundation model with role-aware modeling of endogenous and exogenous modalities, 2026. URL `https://arxiv.org/abs/2609.34842`.

[5] Peng Chen, Siyuan Wang, Shiyan Hu, Xingjian Wu, Yang Shu, Zhongwen Rao, Meng Wang, Yijie Li, Bin Yang, and Chenjuan Guo. Empowering time series analysis with large-scale multimodal pretraining. *arXiv preprint arXiv:2602.05646*, 2026.

[6] Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: Generative pre-trained transformers are large time series models. In *International Conference on Machine Learning*, pages 32369–32399. PMLR, 2024.

[7] Yuxuan Chen, Shanshan Huang, Yunyao Cheng, Peng Chen, Zhongwen Rao, Yang Shu, Bin Yang, Lujia Pan, and Chenjuan Guo. AimTS: Augmented series and image contrastive learning for time series classification. In *ICDE*, 2025.

[8] Shanghua Gao, Teddy Koker, Owen Queen, Thomas Hartvigsen, Theodoros Tsiligkaridis, and Marinka Zitnik. Units: Building a unified time series model. *arXiv*, 2024. URL `https://arxiv.org/pdf/2403.00131.pdf`.

[9] Qichao Shentu, Beibu Li, Kai Zhao, Yang Shu, Zhongwen Rao, Lujia Pan, Bin Yang, and Chenjuan Guo. Towards a general time series anomaly detector with adaptive bottlenecks and dual adversarial decoders. *arXiv preprint arXiv:2405.15273*, 2024.

[10] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. *arXiv preprint arXiv:2402.03885*, 2024.

[11] Yihang Wang, Yuying Qiu, Yang Shu, Zhongwen Rao, Lujia Pan, Bin Yang, and Guo Chenjuan. Lightgts: A lightweight general time series forecasting model. In *ICML*, 2025.

[12] Qingsong Wen, Linxiao Yang, Tian Zhou, and Liang Sun. Robust time series analysis and applications: An industrial perspective. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 4836–4837, 2022.

[13] Yiyuan Yang, Rongshang Li, Qiquan Shi, Xijun Li, Gang Hu, Xing Li, and Mingxuan Yuan. Sgdp: A stream-graph neural network based data prefetcher. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2023.

[14] Tung Kieu, Bin Yang, and Christian S Jensen. Outlier detection for multidimensional time series using deep neural networks. In *2018 19th IEEE international conference on mobile data management (MDM)*, pages 125–134, 2018.

[15] Aditya P Mathur and Nils Ole Tippenhauer. Swat: A water treatment testbed for research and training on ics security. In *CySWater*, pages 31–36, 2016.

[16] Yuwei Cui, Chetan Surpur, Subutai Ahmad, and Jeff Hawkins. A comparative study of htm and other neural network models for online sequence learning with streaming data. In *IJCNN*, pages 1530–1538, 2016.

[17] Alexander von Birgelen and Oliver Niggemann. Anomaly detection and localization for cyber-physical production systems with self-organizing maps. *IMPROVE-Innovative Modelling Approaches for Production Systems to Raise Validatable Efficiency: Intelligent Methods for the Factory of the Future*, pages 55–71, 2018.

[18] Xiangfei Qiu, Xingjian Wu, Yan Lin, Chenjuan Guo, Jilin Hu, and Bin Yang. DUET: Dual clustering enhanced multivariate time series forecasting. In *SIGKDD*, pages 1185–1196, 2025.

[19] Xingjian Wu, Xiangfei Qiu, Zhengyu Li, Yihang Wang, Jilin Hu, Chenjuan Guo, Hui Xiong, and Bin Yang. CATCH: Channel-aware multivariate time series anomaly detection via frequency patching. In *ICLR*, 2025.

[20] Xiangfei Qiu, Yuhan Zhu, Zhengyu Li, Xingjian Wu, Bin Yang, and Jilin Hu. DAG: A dual correlation network for time series forecasting with exogenous variables. In *ICML*, 2026.

[21] Xingjian Wu, Xiangfei Qiu, Hongfan Gao, Jilin Hu, Bin Yang, and Chenjuan Guo. K^2VAE: A koopman-kalman enhanced variational autoencoder for probabilistic time series forecasting. In *ICML*, 2025.

[22] Hanyin Cheng, Xingjian Wu, Xiangfei Qiu, Yang Shu, Bin Yang, and Chenjuan Guo. CCD: Capturing cross-correlations with deformable convolutional networks for multivariate time series forecasting. In *KDD*, 2026.

[23] Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo, Aoying Zhou, Christian S. Jensen, Zhenli Sheng, and Bin Yang. TFB: Towards comprehensive and fair benchmarking of time series forecasting methods. In *Proc. VLDB Endow.*, pages 2363–2377, 2024.

[24] Zhe Li, Xiangfei Qiu, Peng Chen, Yihang Wang, Hanyin Cheng, Yang Shu, Jilin Hu, Chenjuan Guo, Aoying Zhou, Christian S. Jensen, and Bin Yang. TSFM-Bench: Comprehensive and unified benchmarking of foundation models for time series forecasting. In *SIGKDD*, 2025.

[25] Xiangfei Qiu, Xingjian Wu, Hanyin Cheng, Xvyuan Liu, Chenjuan Guo, Jilin Hu, and Bin Yang. DBLoss: Decomposition-based loss function for time series forecasting. In *NeurIPS*, 2025.

[26] Hanyin Cheng, Jingrong Zhou, Yang Shu, and Chenjuan Guo. KITE: Knowledge-guided probabilistic modeling for time series forecasting with exogenous variables. In *ICML*, 2026.

[27] Peng Chen, Yingying Zhang, Yunyao Cheng, Yang Shu, Yihang Wang, Qingsong Wen, Bin Yang, and Chenjuan Guo. Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting. In *ICLR*, 2024.

[28] Jintao Yang, Meng Wang, Chenjuan Guo, and Bin Yang. Rubato: Signature attention for irregular multivariate time series forecasting. In *NeurIPS*, 2026.

[29] Peng Chen, Zhihao Zhuang, Hongzhou Chen, Junhao Huang, Aiping Yang, Mengsen Wu, Yiding Liu, Xilin Dai, and Zewei Dong. Beyond numerical time series: A unified benchmark for multimodal forecasting with heterogeneous context. *arXiv preprint arXiv:2609.15087*, 2026.

[30] Xingze Zheng, Hanyin Cheng, Siyuan Wang, Yiting Hao, Peng Chen, Yuan Jun, and Yang Shu. Scan: Enhance time series anomaly detection via multi-scale neighborhood-centered clustering, 2026. URL `https://arxiv.org/abs/2606.19255`.

[31] Hongfan Gao, Wangmeng Shen, Xiangfei Qiu, Ronghui Xu, Bin Yang, and Jilin Hu. Ssd-ts: Exploring the potential of linear state space models for diffusion models in time series imputation. In *SIGKDD*, 2025.

[32] Wangmeng Shen, Hongfan Gao, Qingsong Zhong, Dingli Xu, and Jilin Hu. FSDI: frequency-shaped diffusion for time-series imputation. In *WWW 2026*, pages 5310–5321, 2026.

[33] Xingjian Wu, Junkai Lu, Zhengyu Li, Xiangfei Qiu, Jilin Hu, Chenjuan Guo, Christian S Jensen, and Bin Yang. TimeART: Towards agentic time series reasoning via tool-augmentation. *arXiv preprint arXiv:2601.13653*, 2026.

[34] Junkai Lu, Peng Chen, Xingjian Wu, Yang Shu, Chenjuan Guo, Christian S. Jensen, and Bin Yang. PATRA: Pattern-aware alignment and balanced reasoning for time series question answering. In *Forty-third International Conference on Machine Learning*, 2026. URL `https://openreview.net/forum?id=Du8BMHfNzp`.

[35] 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*, pages 93–104, 2000.

[36] Markus Goldstein and Andreas Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI-2012: poster and demo track*, 1:59–63, 2012.

[37] Sahand Hariri, Matias Carrasco Kind, and Robert J Brunner. Extended isolation forest. *IEEE transactions on knowledge and data engineering*, 33(4):1479–1489, 2019.

[38] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *Proceedings of the 2008 IEEE international conference on data mining*, pages 413–422, 2008.

[39] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD international conference on management of data*, pages 427–438, 2000.

[40] Zhijie Zhong, Zhiwen Yu, Xing Xi, Yue Xu, Wenming Cao, Yiyuan Yang, Kaixiang Yang, and Jane You. Simad: A simple dissimilarity-based approach for time-series anomaly detection. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

[41] Qinghua Liu, Paul Boniol, Themis Palpanas, and John Paparrizos. Time-series anomaly detection: Overview and new trends. *Proc. VLDB Endow.*, 17(12):4229–4232, 2024.

[42] Qinghua Liu and John Paparrizos. The elephant in the room: Towards a reliable time-series anomaly detection benchmark. In *NeurIPS*, 2024.

[43] Lingpei Zhang, Qingming Li, Yong Yang, Jiahao Chen, Rui Zeng, Chenyang Lyu, and Shouling Ji. Contextual and seasonal LSTMs for time series anomaly detection. In *The Fourteenth International Conference on Learning Representations*, 2026.

[44] Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4027–4035, 2021.

[45] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *ICLR*, 2023.

[46] Donghao Luo and Xue Wang. Moderntcn: A modern pure convolution structure for general time series analysis. In *The Twelfth International Conference on Learning Representations*, 2024.

[47] Shreshth Tuli, Giuliano Casale, and Nicholas R Jennings. Tranad: Deep transformer networks for anomaly detection in multivariate time series data. *arXiv preprint arXiv:2201.07284*, 2022.

[48] Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Anomaly Transformer: Time series anomaly detection with association discrepancy. In *International Conference on Learning Representations*, 2021.

[49] Yiyuan Yang, Chaoli Zhang, Tian Zhou, Qingsong Wen, and Liang Sun. DCdetector: Dual attention contrastive representation learning for time series anomaly detection. In *SIGKDD*, pages 3033–3045, 2023.

[50] Xiaojun Guo, Yifei Wang, Tianqi Du, and Yisen Wang. Contranorm: A contrastive learning perspective on oversmoothing and beyond. In *The Eleventh International Conference on Learning Representations*, 2023.

[51] Beibu Li, Qichao Shentu, Yang Shu, Hui Zhang, Ming Li, Ning Jin, Bin Yang, and Chenjuan Guo. Crossad: Time series anomaly detection with cross-scale associations and cross-window modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[52] Junxian Li, Keke Huang, Dehao Wu, Yishun Liu, Chunhua Yang, and Weihua Gui. Hybrid variable dictionary learning for monitoring continuous and discrete variables in manufacturing processes. *Control Engineering Practice*, 149:105970, 2024.

[53] Junhao Chen, Chunhui Zhao, Pengyu Song, and Min Xie. Unified low-dimensional subspace analysis of continuous and binary variables for industrial process monitoring. *IEEE Transactions on Cybernetics*, 2025.

[54] Yihang Wang, Yuying Qiu, Peng Chen, Kai Zhao, Yang Shu, Zhongwen Rao, Lujia Pan, Bin Yang, and Chenjuan Guo. Rose: Register assisted general time series forecasting with decomposed frequency learning. *arXiv preprint arXiv:2405.17478*, 2024.

[55] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning*, 2024.

[56] Peng Chen, Yihang Wang, Yang Shu, Yunyao Cheng, Kai Zhao, Zhongwen Rao, Lujia Pan, Bin Yang, and Chenjuan Guo. Cc-time: Cross-model and cross-modality time series forecasting. *arXiv preprint arXiv:2508.12235*, 2025.

[57] Abdelhakim Benechehab, Vasilii Feofanov, Giuseppe Paolo, Albert Thomas, Maurizio Filippone, and Balázs Kégl. Adapts: Adapting univariate foundation models to probabilistic multivariate time series forecasting. *arXiv preprint arXiv:2502.10235*, 2025.

[58] Zhongzheng Qiao, Chenghao Liu, Yiming Zhang, Ming Jin, Quang Pham, Qingsong Wen, PN Suganthan, Xudong Jiang, and Savitha Ramasamy. Multi-scale finetuning for encoder-based time series foundation models. *arXiv preprint arXiv:2506.14087*, 2025.

[59] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 255–262, 2016.

[60] Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

[61] Xiaopeng Li, Shasha Li, Shezheng Song, Huijun Liu, Bin Ji, Xi Wang, Jun Ma, Jie Yu, Xiaodong Liu, Jing Wang, et al. Swea: Updating factual knowledge in large language models via subject word embedding altering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24494–24502, 2025.

[62] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

[63] Pat Lovie. Coefficient of variation. *Encyclopedia of statistics in behavioral science*, 2005.

[64] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[65] Chun-Guo Li. Learn hybrid prototypes for multivariate time series anomaly detection. In *The Thirteenth International Conference on Learning Representations*, 2025.

[66] Gilbert W Stewart. On the early history of the singular value decomposition. *SIAM review*, 35 (4):551–566, 1993.

[67] HyunGi Kim, Jisoo Mok, Dongjun Lee, Jaihyun Lew, Sungjae Kim, and Sungroh Yoon. Causality-aware contrastive learning for robust multivariate time-series anomaly detection. In *Forty-second International Conference on Machine Learning*, 2025.

[68] HyunGi Kim, Siwon Kim, Seonwoo Min, and Byunghan Lee. Contrastive time-series anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, 36(10):5053–5065, 2024.

[69] Yuchen Fang, Jiandong Xie, Yan Zhao, Lu Chen, Yunjun Gao, and Kai Zheng. Temporal-frequency masked autoencoders for time series anomaly detection. In *International Conference on Data Engineering, ICDE 2024*, pages 1228–1241, 2024.

[70] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *SIGKDD*, pages 387–395, 2018.

[71] John Paparrizos, Paul Boniol, Themis Palpanas, Ruey S Tsay, Aaron Elmore, and Michael J Franklin. Volume Under the Surface: A New Accuracy Evaluation Measure for Time-Series Anomaly Detection. *Proc. VLDB Endow.*, 15(11):2774–2787, 2022.

[72] Xiangfei Qiu, Zhe Li, Wanghui Qiu, Shiyan Hu, Lekui Zhou, Xingjian Wu, Zhengyu Li, Chenjuan Guo, Aoying Zhou, Zhenli Sheng, Jilin Hu, Christian S. Jensen, and Bin Yang. TAB: Unified benchmarking of time series anomaly detection methods. In *Proc. VLDB Endow.*, pages 2775–2789, 2025.

[73] Nesime Tatbul, Tae Jun Lee, Stan Zdonik, Mejbah Alam, and Justin Gottschlich. Precision and recall for time series. *NeurIPS*, 31, 2018.

[74] Alexis Huet, Jose Manuel Navarro, and Dario Rossi. Local evaluation of time series anomaly detection algorithms. In *SIGKDD*, pages 635–645, 2022.

[75] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the international conference on machine learning*, pages 233–240, 2006.

[76] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[77] Ahmed Abdulaal, Zhuanghua Liu, and Tomer Lancewicki. Practical approach to asynchronous multivariate time series anomaly detection and localization. In *SIGKDD*, pages 2485–2494, 2021.

[78] Cheng Feng and Pengwei Tian. Time series anomaly detection for cyber-physical systems via neural system identification and bayesian filtering. In *SIGKDD*, pages 2858–2867, 2021.

[79] Susik Yoon, Jae-Gil Lee, and Byung Suk Lee. Ultrafast local outlier detection from a data stream with stationary region skipping. In *SIGKDD*, pages 1181–1191, 2020.

A Experimental Details

A.1 Data Analysis

In the vast majority of cases, explicit input type information is unnecessary for the data processing. A significant difference between numerical variables and state variables in real-world scenarios lies in the number of their possible values. State variables are typically constrained to a limited number of outcomes, usually fewer than fifty. Conversely, numerical variables can assume a vast range of values, often numbering in the hundreds or thousands. This clear divergence in the size of their value space is sufficient to distinguish one type from the other.

Table 6 presents a statistical summary of the value set cardinality for both numerical variables and state variables.

Table 6: Data Analysis.

Dataset	**Numerical Variables**				**State Variables**			
Metric	**Maximum**	**Minimum**	**Median**	**Average**	**Maximum**	**Minimum**	**Median**	**Average**
MSL	35103	35103	35103	35103.0	2	2	2.0	2.0
SMAP	75332	75332	75332	75332.0	2	2	2	2.0
Genesis	4074	512	2208	2250.8	21	2	2	3.3
NYC	11790	11790	11790	11790.0	48	2	7	19.0
SWaT	14626	116	2144	4002.6	37	2	2.0	3.9

A.2 Time Series Foundation Models for Anomaly Detection

In the realm of time series analysis, numerous models have surfaced in recent years. We choose models designed for multiple tasks or specialized for anomaly detection, including task-specific model DADA [9] and task-general models: UniTS [8], Moment [10], Timer [6]. The specific descriptions for each of these models are listed in Table 7.

Table 7: Descriptions of Time Series Foundation Models for Anomaly Detection in experiments.

Models	**Descriptions**
DADA	DADA is a general time series anomaly detector pre-trained on multi-domain data, utilizing adaptive bottlenecks and dual adversarial decoders. It detects anomalies by learning to distinguish normal from abnormal patterns.
UniTS	UniTS is a unified, multi-task transformer model that uses task tokenization to handle a variety of time series tasks like forecasting, classification, imputation, and anomaly detection within a single framework.
Moment	Moment is a transformer system pre-trained on a masked time series task. It reconstructs masked portions of time series for tasks like forecasting, classification, anomaly detection, and imputation.
Timer	Timer is a GPT-style autoregressive model for time series analysis, predicting the next token in single-series sequences. It supports tasks like forecasting, imputation, and anomaly detection across different time series.

A.3 Metrics for Time Series Anomaly Detection

The metrics used for evaluation can be divided into two categories: Threshold-relevant and Threshold-irrelevant. Threshold-relevant metrics depend on threshold parameters to convert anomaly scores into anomaly labels. Threshold-irrelevant metrics, on the other hand, evaluate the performance of the model based on the raw anomaly scores it generates. Please refer to Table 8 for an overview of these metrics.

Table 8: Overview of evaluation metrics.

Category	**Metric**	**Abbreviation**	**Short summary**
Threshold-relevant	Accuracy	Acc	Measures the proportion of correct predictions among the total number of predictions.
	Range-Precision Range-Recall Range-F1-score	R-P [73] R-R [73] R-F1 [73]	These metrics' definitions consider several factors: the ratio of detected anomaly subsequences to the total number of anomalies, the ratio of detected point outliers to total point outliers, the relative position of true positives within each anomaly subsequence, and the number of fragmented prediction regions corresponding to one real anomaly subsequence.
	Affiliated-Precision Affiliated-Recall Affiliated-F1-score	Aff-P [74] Aff-R [74] Aff-F1 [74]	These metrics' definitions are the extension of the classical precision/recall/f1-score for time series anomaly detection that is local (each ground truth event is considered separately), parameter-free, and applicable generically on both point and subsequence anomalies. Besides the construction of these metrics makes them both theoretically principled and practically useful.
Threshold-irrelevant	AUC-PR	A-P [75]	Measures the area under the curve corresponding to Recall on the x-axis and Precision on the y-axis at various threshold settings.
	AUC-ROC	A-R [76]	Measures the area under the curve corresponding to FPR on the x-axis and TPR on the y-axis at various threshold settings.
	R-AUC-PR R-AUC-ROC	R-A-P [71] R-A-R [71]	They mitigate the issue that AUC-PR and AUC-ROC are designed for point-based anomaly detection, where each point is assigned equal weight in calculating the overall AUC. This makes them unsuitable for evaluating subsequence anomalies.
	VUS-PR VUS-ROC	V-PR [71] V-ROC [71]	VUS is an extension of the ROC and PR curves. It introduces a buffer region at the outliers' boundaries, thereby accommodating the false tolerance of labeling in the ground truth and assigning higher anomaly scores near the outlier boundaries.

B More Analysis on STAR

B.1 Applicability for dataset

While most industrial scenarios feature a variety of *state variables*, there are also special or constrained settings where such variables are unavailable. To further extend the applicability of STAR, we explored the approach of proactively constructing pseudo-covariates to serve as *state variables* for processing by the STAR module. Specifically, numerical variables often inherently carry state information. For instance, temperature indicates whether the system is in a high or low-temperature state, while the pH value reflects the precondition of an acidic or alkaline environment. We leverage this property by discretizing these numerical variables into discrete states to construct our pseudo-covariates. We conducted this experiment on the PSM [77] , PUMP [78], and ECG [79] datasets, and the results are presented in Table 9. Rows marked with an asterisk (*) denote the use of pseudo-covariates. The results indicate that in most cases, the incorporation of *state variables* still leads to a decrease in performance. Although the pseudo-covariates lack a well-defined physical meaning, STAR is nevertheless able to improve the performance of TSFMs to a certain degree.

Table 9: Evaluate STAR on datasets containing solely numerical variables.

Dataset	PSM			PUMP			ECG		
Metric	A-R	V-R	V-P	A-R	V-R	V-P	A-R	V-R	V-P
DADA	0.606	0.581	0.403	0.775	0.778	0.226	0.839	0.807	0.508
DADA*	0.621	0.588	0.391	0.764	0.777	0.223	0.828	0.801	0.495
+STAR*	**0.636**	**0.592**	**0.412**	**0.807**	**0.807**	**0.236**	**0.844**	**0.811**	**0.514**
UniTS	0.582	0.573	0.377	0.415	0.573	0.211	0.872	0.840	0.522
UniTS*	0.548	0.544	0.354	0.384	0.536	0.195	0.863	0.835	0.516
+STAR*	**0.607**	**0.602**	**0.401**	**0.436**	**0.598**	**0.217**	**0.884**	**0.848**	**0.537**

B.2 Case study

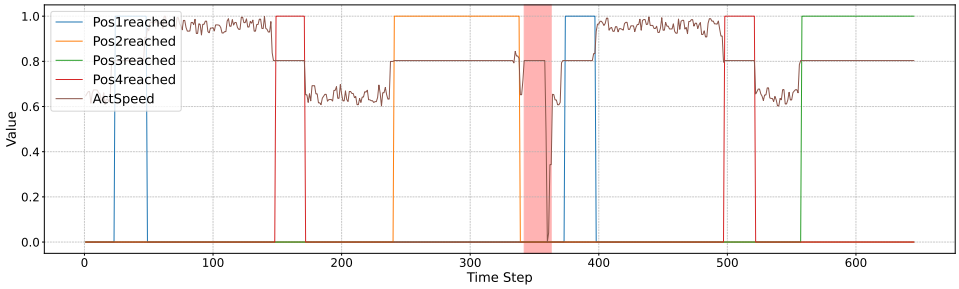
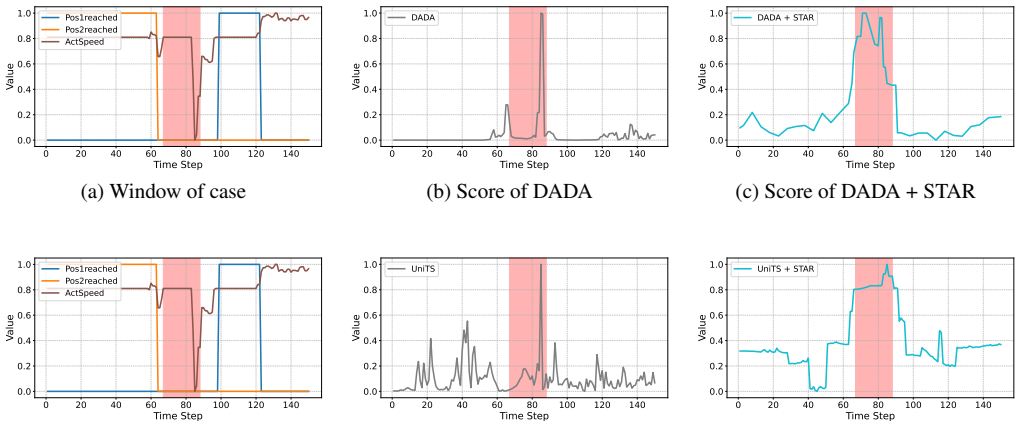


Figure 6: Case series in Genesis.



(a) Window of case (b) Score of DADA (c) Score of DADA + STAR

(d) Window of case (e) Score of UniTS (f) Score of UniTS + STAR

Figure 7: Anomaly score of backbones and STAR in case from Genesis.

To further analyze how STAR works, we selected and visualized cases from real-world datasets.

(1) **Genesis** Figure 6 illustrates that the position state variables (Pos1reached, Pos2reached, Pos3reached, and Pos4reached) have a condition-based influence on the actual motor speed (ActSpeed). Under normal circumstances, the actual motor speed exhibits a stable pattern when the system is in a state associated with reaching a specific position (PosXreached = 1). However, when the system has not reached any specific position, the motor speed displays an unstable pattern (PosXreached = 0).

The anomaly presented in the case is that the actual motor speed exhibits a stable pattern even when the system is in a state of not having reached any specific position. This constitutes a state-value mismatch anomaly. As shown in Figure 7, although TSFMs are highly sensitive to the abrupt change at the final timestamps of the anomaly, they fail to detect the preceding portion. With the integration of STAR, the model gains heightened sensitivity to the state-value mismatch anomaly and successfully captures this anomaly case.

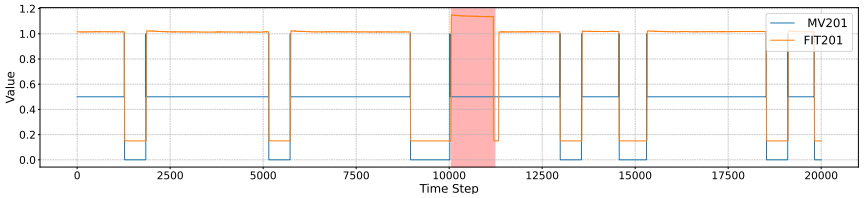
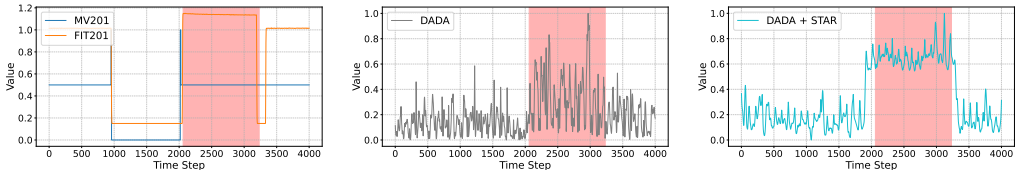
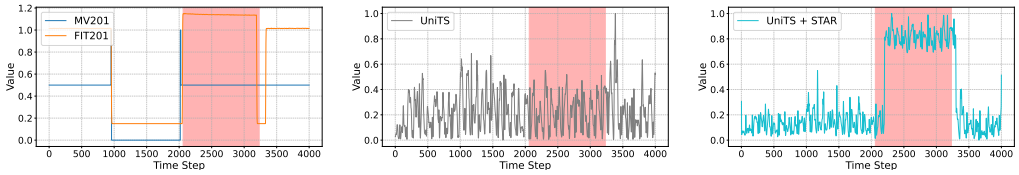


Figure 8: Case series in SWaT.



(a) Window of case (b) Score of DADA (c) Score of DADA + STAR



(d) Window of case (e) Score of UniTS (f) Score of UniTS + STAR

Figure 9: Anomaly score of backbones and STAR in case from SWaT.

(2) **SWaT** As shown in Figure 8, the state variable of motorized valve switch (MV201) also has a condition-based influence on the water flow (FIT201). Under normal conditions, the behavior is as follows: when the valve is open (MV201=0.5), the water flow remains stable at a high level; when the valve is closed (MV201=0), the flow stabilizes at a low level; and during a transitional state (MV201=1), the flow rapidly increases or decreases.

The anomaly presented in the case is twofold. On the one hand, the water level is excessively high. On the other hand, the flow begins to decrease rapidly even though the valve is not in a transitional state (MV201=1), remaining fully open (MV201=0.5). Although DADA successfully identified the excessively high water level, it failed to detect the anomaly of the rapid decrease. This is likely because the variable itself frequently exhibits patterns of rapid decline over short periods. STAR successfully captures this anomaly by considering the degree of matching between the state and the numerical value.

B.3 Visualisation of bottleneck across different state

To analyze the bottleneck activations under different states, we visualize both the raw time series data and the corresponding mask vector. This mask vector is conditioned on state variables and applied to

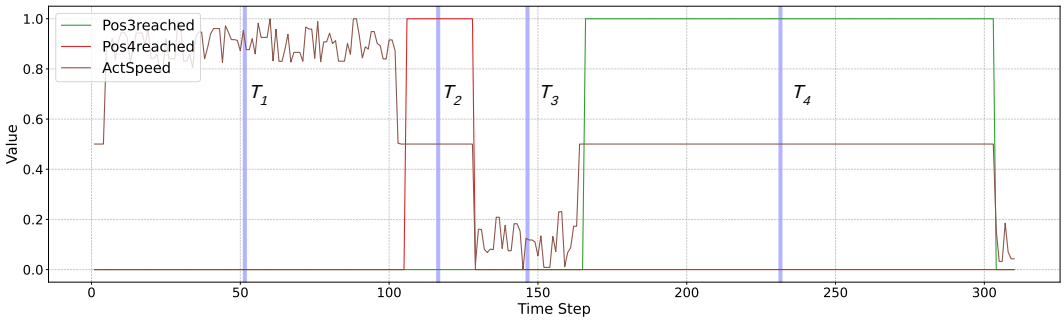


Figure 10: Case series in Genesis.

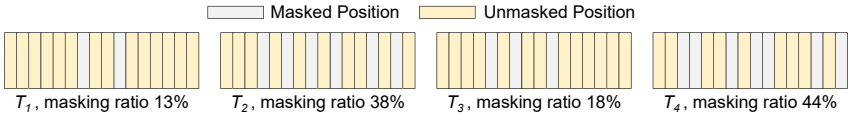


Figure 11: Visualization of the mask vector of $\boldsymbol{R}$.

matrix $\boldsymbol{R}$, which is inserted between matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ in the Conditional Bottleneck Adapter to adaptively modulate the bottleneck, as defined in Equation 9.

Figure 10 illustrates the raw time series, highlighting the relationship between numerical and state variables in the Genesis dataset. When all state variables (Pos3reached and Pos4reached) are zero, the numerical variable (ActSpeed) exhibits complex and rapid fluctuations; otherwise, it displays a stable pattern. We specifically selected representative timestamps T_1 and T_3 to exemplify the rapid fluctuation pattern, and T_2 and T_4 for the relatively stable pattern, as indicated by the purple markers in the figure.

Figure 11 illustrates the mask vector of matrix $\boldsymbol{R}$ corresponding to timestamps T_1, T_2, T_3, and T_4, along with its masking ratio. This masking ratio reflects the bottleneck size of the weights in the Conditional Bottleneck Adapter.

Timestamps T_1 and T_3 exemplify the rapid fluctuation pattern. In these instances, the numerical variable is relatively complex and requires more temporal information for reconstruction. Consequently, the masking ratio of the corresponding matrix $\boldsymbol{R}$ is lower, indicating a larger bottleneck size. In contrast, T_2 and T_4 represent the stable pattern, where the numerical variable is relatively simple and requires less temporal information. This results in a higher masking ratio for $\boldsymbol{R}$, signifying a smaller bottleneck size.

This phenomenon justifies our strategy of dynamically adjusting the bottleneck size according to the system state.

C Limitations

Despite the effectiveness of STAR in enhancing time series foundation models through state-aware adaptation, it still has a few limitations:

- STAR primarily operates in the fine-tuning stage, where the backbone is frozen and only the STAR modules are optimized. Consequently, when sufficient fine-tuning data are unavailable, its detection accuracy may be suboptimal.
- STAR detects state-related anomalies mainly through patch-level matching between numerical variables and state variables. However, the patch size is determined by the backbone's pre-training configuration, which limits the flexibility of adaptation during fine-tuning.

D Impact Statement

This paper presents work aimed at advancing the field of Machine Learning. There are many potential societal consequences of our research, none of which we feel must be specifically highlighted here.

Furthermore, while the Large Language Models were used to polish some grammar during the writing process, all content remains original rather than generated, and there are no ethical issues to report.

E The Use of Large Language Models

LLMs were used only for minor writing assistance, such as improving grammar and wording in the manuscript. This auxiliary usage did not affect the method design, experimental results, or scientific conclusions of the paper.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 describe the motivation, the three components of STAR, and the scope of the empirical claims, and these claims are matched by the method and experiments in Sections 3–4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We provide the discussion of the limitations in Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper is an empirical methods paper and does not present theorem-level theoretical results or formal proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The performance of STAR and the datasets used in our work are real, and all experimental results can be reproduced. We have released our model code at https://github.com/decisionintelligence/STAR.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We have released the code and data for the experimental results at: https://github.com/decisionintelligence/STAR

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The manuscript specifies datasets, evaluation metrics, backbone models, and some hyperparameter studies in Section 4.1 and the Appendix A, but it does not fully report all optimization and implementation details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [N/A]

   Justification: Following the experimental protocol of the previous work, we set a fixed random seed to ensure deterministic results across all runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: We didn't include the compute resources.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work studies anomaly detection on benchmark time-series datasets, does not involve human subjects experimentation, and is intended to improve reliability and monitoring in practical systems.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We provide it in Appendix D.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a high-risk generative model or scraped dataset of the kind contemplated by this question.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All external resources including open-source code, public datasets, and pre-trained models adopted in this work are properly cited in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new assets introduced in the paper are well-documented, and the documentation is provided alongside the assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Our research does not involve these concerns.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The paper does not involve LLMs as any important, original, or non-standard components. We discuss this in Appendix E.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.