TENET: Time-point Encoding Network for Multivariate Time Series Anomaly Detection

Dongchan Cho* **Juwon Hwang** **Keumyeong Kang** **Jiho Han** **Namsoon Jung**
Industrial AI Lab, SimPlatform Co. Ltd. Affiliate Research Institute
{cdc0213, jihohan, jwhwang, chmin, mskim, nsj}@simplatform.com

Abstract

In unsupervised multivariate time series anomaly detection (MTSAD), local variation can inflate reconstruction and forecasting errors, making observation-space discrepancy an unreliable indicator of abnormal system behavior. Since latent-space scoring can mitigate this limitation, we reframe one-class MTSAD as current-point system-state compatibility estimation and propose **TENET**, the **T**ime-point **E**ncoding **Net**work for Multivariate Time Series Anomaly Detection. We define the current-point system state as a latent representation of the target time point, constructed from variable-level temporal context in the input window. TENET constructs this state through a temporal layer that retrieves window context relative to the current point, encodes time with current-point-centered embeddings, and adaptively mixes the retrieved context with the current representation. With this design, TENET treats the input window not as the object to reconstruct, forecast, summarize, or score, but as context for constructing a latent representation evaluated with a fixed standard-normal negative log-likelihood score. Across benchmark datasets, TENET achieves state-of-the-art performance. Additional experiments further show that TENET remains effective under alternative one-class objectives and outperforms substitutions based on generic sequence encoders, suggesting that its advantage comes from decision-aligned representation construction rather than from a specialized scoring rule. These results indicate that one-class MTSAD can be strong when the latent representation is designed around the current time point.

1 Introduction

Detecting anomalies in multivariate time series requires reasoning about the nonlinear dependencies that characterize normal system behavior [53]. Because real-world anomalies constitute an open set and are often rare or previously unseen [12, 29, 36, 37], supervised detection is often impractical. As a result, multivariate time series anomaly detection (MTSAD) has largely shifted toward unsupervised learning, where the detector must infer normality from normal operation alone [6, 7, 18, 42, 49]. Observation-space discrepancy has therefore become a natural and often effective anomaly signal in unsupervised MTSAD, as reflected in methods based on reconstruction residuals [7, 22, 25, 35, 42, 44], forecasting errors [3, 6, 9, 23], or local temporal inconsistencies [52].

However, using observation-space discrepancy as an anomaly score relies on a strong surrogate assumption [4, 48]. Reconstruction and forecasting errors are meaningful only when the observation-space target can be reproduced or predicted with sufficient fidelity and when the remaining mismatch mainly reflects abnormal system behavior. This assumption can be fragile in real-world multivariate data, where normal signals may contain local variation, high-frequency fluctuations, or discontinuous changes that are difficult to reconstruct or predict even when the system remains normal. In such cases, a large observation-space error may reflect modeling error in the reconstruction or prediction task

*Corresponding author.

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

rather than an abnormal state. This limitation motivates latent one-class scoring, where abnormality is assessed through compatibility with a normal state space rather than through raw mismatch alone.

While one-class MTSAD addresses this surrogate problem by scoring latent normality, its semantic advantage has not been fully exploited. Existing latent-space anomaly detectors have largely advanced by changing the objective used to shape or score the normal space, such as hypersphere or center compactness [34, 40], density or energy estimation [56], and probabilistic likelihood-style scoring [42], while leaving the construction of the scored representation comparatively underexamined. This matters because a one-class objective can only evaluate the representation it is given. If that representation is obtained by generic sequence compression or window-level summarization, the detector may still be scoring an object that is misaligned with the actual decision target.

In point-wise MTSAD, the target is the current time point, not the window itself. We use *state* to denote the latent representation evaluated for normality at that target point. Because the input window contains both target-relevant and target-irrelevant history, a window-level summary can dilute the information needed for the current decision. The fundamental question is therefore not only how to score normality in latent space, but what state should be constructed before scoring. We argue that the window should be used as context for the current point, so the encoder actively retains history that helps explain this state rather than uniformly accumulating the past or preserving information for reconstruction and forecasting. At the multivariate level, this current-centered temporal context should be formed separately for each variable and then composed into a *current-point system state* before scoring. This shifts the bottleneck in one-class MTSAD from designing stronger normality objectives to constructing a decision-aligned state that makes one-class scoring meaningful.

Motivated by this view, we reformulate one-class MTSAD as estimating the compatibility of a current-point system state with normal behavior and propose **T**ime-point **E**ncoding **Net**work for Multivariate Time Series Anomaly Detection (**TENET**). TENET is a current-point-centric encoder that explicitly constructs the current-point system state before compatibility assessment. Its temporal layer uses the current point as an anchor to retrieve relevant recent history for each variable, after which inter-variable dependencies are modeled over the resulting current-point state tokens and aggregated into the current-point system state. This compatibility is then assessed during both training and inference, instantiated in our model as a fixed standard-normal NLL compatibility score.

Our main contributions are as follows:

- We reformulate unsupervised one-class MTSAD as current-point system-state compatibility estimation and identify decision-aligned current-point system-state construction, rather than increasingly complex raw-space discrepancy modeling, as the central challenge.
- We propose TENET, a current-point-centric encoder that constructs the current-point system state by retrieving temporally relevant history with respect to the present and composing inter-variable dependencies before compatibility assessment.
- We show that TENET achieves state-of-the-art results across standard MTSAD benchmarks, while representation diagnostics indicate that its gains come from current-point-centric state construction rather than from the scoring objective alone.

2 Related Work

2.1 Latent-Space One-Class MTSAD

Alongside observation-space discrepancy-based MTSAD, latent-space anomaly detection provides a complementary route that scores learned representations with one-class or probabilistic objectives instead of measuring reconstruction or prediction residuals. Representative examples include Deep SVDD [34], THOC [40], DAGMM [56], and probabilistic latent-variable models such as OmniAnomaly [42]. Despite substantial differences in formulation, these methods share the view that abnormality can be assessed in a learned normal state space rather than directly in the observation space. Accordingly, much of the methodological variation in this line of work has centered on how the latent space is shaped, for example through hypersphere or center-based compactness [34, 40], Gaussian mixture density estimation in a compressed latent space [56], and variational latent-variable modeling [42]. In comparison, our work focuses on the less explicit question of how to construct the latent state that is scored for current-point anomaly decisions in multivariate time series.

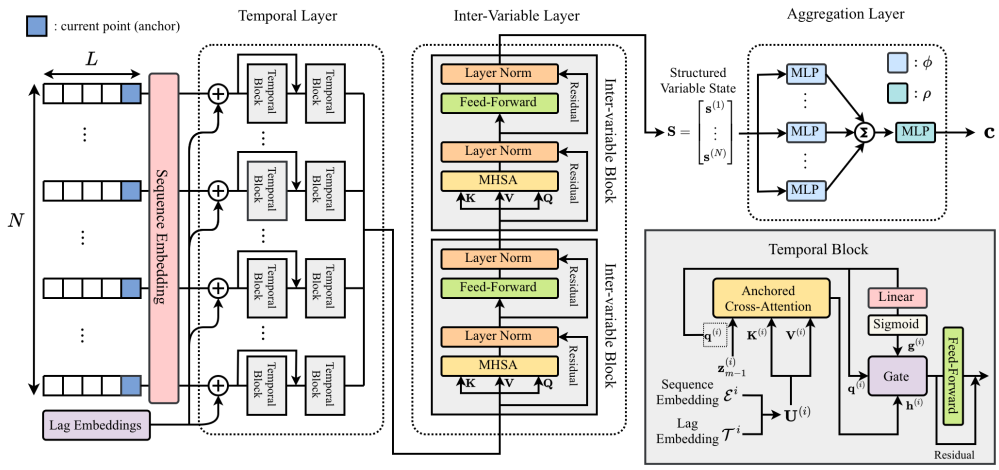

Figure 1: Architecture of TENET. A temporal layer, an inter-variable layer, and an aggregation layer construct the current-point system state from the input window. The inset shows the Temporal block, which uses *anchored cross-attention* to retrieve history relative to the current point.

2.2 Sequence Encoders for MTSAD

Our work is also related to multivariate time series encoders used as backbones for forecasting, reconstruction, classification, and anomaly detection. Common choices include recurrent encoders such as LSTM [26, 43], convolutional and temporal-convolution models such as TCN [5, 25], Transformer-style encoders [22, 28, 45, 54], and more recent state-space models such as Mamba [13]. Recent TSAD models also build on these backbones. xLSTMAD [10] uses an xLSTM encoder–decoder for forecasting or reconstruction, while MemMambaAD [19] combines a Mamba–TCN encoder with memory-guided reconstruction. PaAno [31] instead learns local patch representations and scores their distances to a normal-patch memory bank. These methods encode sequences or patches, whereas TENET uses the current point to query variable-level history before scoring the resulting representation. Particularly relevant to our work are variable-aware designs that preserve variable-level representations before or during multivariate modeling. PatchTST [28] adopts channel-independent temporal encoding, while Crossformer [54], iTransformer [22], and OracleAD [6] further introduce mechanisms for modeling dependencies across variable-level representations. TENET differs by using the input window as variable-level context for constructing the state of the current time point, rather than compressing the entire window into a generic sequence representation.

3 Method

3.1 Problem Setup

We consider unsupervised multivariate time series anomaly detection (MTSAD). Let the observed multivariate time series be denoted by $\mathcal{X} = \{\mathbf{x}_t\}_{t=1}^{T}$, where $\mathbf{x}_t \in \mathbb{R}^N$ is the observation vector at time t over N variables. We assume that the training split contains only normal operation, whereas the test split may contain unknown anomalous events. The detector operates on sliding windows. For each time step $t \geq L$, we form an input window of length L as

$$\mathbf{X}_t = [\mathbf{x}_{t-L+1}, \ldots, \mathbf{x}_t] \in \mathbb{R}^{L \times N}. \tag{1}$$

Within this window, the last observation $\mathbf{x}_t$ is the current point to be evaluated, while the preceding sequence $\mathbf{x}_{t-L+1:t-1}$ provides local temporal evidence. When the context is clear, we omit the subscript t and write $\mathbf{X}$. Our goal is to assign an anomaly score to each current point $\mathbf{x}_t$ based on the compatibility of its latent current-point system state with normal operation. Here, the current-point system state denotes a learned representation for anomaly detection. Rather than treating $\mathbf{X}_t$ itself as the decision object, we use it as context for constructing this state.

3.2 Model Architecture

TENET constructs the current-point system state $\mathbf{c}$ from a multivariate window $\mathbf{X}$ in three stages: a temporal layer f_t, an inter-variable layer f_v, and an aggregation layer f_g. The first two stages produce a structured variable-level representation, while the last stage maps it to the current-point system state used for anomaly scoring. Rather than encoding the entire multivariate window into a generic latent vector in a single step, this staged construction first forms current-point state tokens for individual variables, then composes them across variables, and only afterwards produces the current-point system state $\mathbf{c}$. A detailed design rationale is provided in Appendix A.

Temporal Layer. To reduce cross-variable interference at the temporal encoding stage, we adopt a variable-aware strategy and encode each variable independently [6, 15, 22, 28]. Accordingly, each variable is processed by its own temporal encoder parameterized by θ_i, allowing TENET to preserve variable-specific temporal characteristics before any cross-variable interaction is introduced. For notational simplicity, within each window we re-index local time positions from 1 to L, so that $\mathbf{x}_{1:L}^{(i)}$ denotes the univariate sequence of variable i in $\mathbf{X}$. For each variable i, the temporal layer $f_t(\cdot;\theta_i)$ maps the univariate window $\mathbf{x}_{1:L}^{(i)}$ to a current-point state token. The key design choice is to encode temporal context according to its relevance to the target point, so that the temporal layer constructs the variable state at that point rather than summarizing the window uniformly over time. In this sense, the temporal layer performs *current-point-conditioned history retrieval*: the current point determines which historical information is most relevant for constructing the current-point variable state. Operationally, f_t is implemented as a stack of M Temporal blocks, each of which updates the current-point anchor by retrieving history from the local window.

We first map $\mathbf{x}_{1:L}^{(i)}$ to value embeddings $\mathcal{E}^{(i)} \in \mathbb{R}^{L\times d}$. To encode temporal order relative to the current point, we use anchor-centered lag embeddings [39]. For each position $\ell \in \{1,\dots,L\}$, we define the lag $\tau_\ell = L - \ell$, so that $\tau_L = 0$ corresponds to the current point and larger τ_ℓ indicates farther history. We then obtain temporal embeddings $\boldsymbol{\mathcal{T}}^{(i)} \in \mathbb{R}^{L\times d_t}$, where the lag embedding at position ℓ is denoted by $\boldsymbol{\mathcal{T}}_\ell^{(i)} = \mathbf{r}_{\tau_\ell}$ with $\mathbf{r}_{\tau_\ell} \in \mathbb{R}^{d_t}$. The value and lag embeddings are concatenated to form the variable-specific tokens $\mathbf{U}^{(i)} = \left[\mathcal{E}^{(i)}; \boldsymbol{\mathcal{T}}^{(i)}\right] \in \mathbb{R}^{L\times(d+d_t)}$.

The temporal layer is current-point-centered: it starts from the current point and retrieves recent history only to the extent that the history helps explain the present state. Accordingly, we initialize the evolving anchor state by the embedding of the last time point, $\mathbf{z}_0^{(i)} = \mathcal{E}_L^{(i)} \in \mathbb{R}^d$. At Temporal block m, the core operation is an *anchored cross-attention* module: the query is derived from the current anchor state, while keys and values are obtained from the full window after token-wise normalization:

$$\mathbf{q}^{(i)} = \mathrm{LN}_q\Big(\mathbf{z}_{m-1}^{(i)}\Big)\,\mathbf{W}_Q, \quad \mathbf{K}^{(i)} = \mathrm{LN}_{kv}\Big(\mathbf{U}^{(i)}\Big)\,\mathbf{W}_K, \quad \mathbf{V}^{(i)} = \mathrm{LN}_{kv}\Big(\mathbf{U}^{(i)}\Big)\,\mathbf{W}_V. \tag{2}$$

For head $r \in \{1,\dots,H_a\}$ with per-head dimension $d_h = d/H_a$, the attention weights and retrieved context are

$$\boldsymbol{\alpha}^{(i,r)} = \mathrm{softmax}\left(\frac{\mathbf{q}^{(i,r)}(\mathbf{K}^{(i,r)})^\top}{\sqrt{d_h}}\right) \in \mathbb{R}^{1\times L}, \qquad \mathbf{h}^{(i,r)} = \boldsymbol{\alpha}^{(i,r)}\mathbf{V}^{(i,r)} \in \mathbb{R}^{1\times d_h}, \tag{3}$$

and the multi-head output is $\mathbf{h}^{(i)} = \left[\mathbf{h}^{(i,1)} \;\; \cdots \;\; \mathbf{h}^{(i,H_a)}\right]\mathbf{W}_O \in \mathbb{R}^d$. The retrieved context is merged with the anchor state through a *gating mechanism*:

$$\begin{aligned} \mathbf{g}^{(i)} &= \sigma\big(\mathbf{q}^{(i)}\mathbf{W}_g + \mathbf{b}_g\big), \\ \tilde{\mathbf{z}}_m^{(i)} &= \mathbf{g}^{(i)} \odot \mathbf{q}^{(i)} + (1-\mathbf{g}^{(i)}) \odot \mathbf{h}^{(i)}. \end{aligned} \tag{4}$$

Here, $\sigma(u) = 1/(1+e^{-u})$ is the element-wise sigmoid, and $\mathbf{g}^{(i)} \in (0,1)^d$ controls how much the model relies on the anchor state versus retrieved history. The resulting representation is then refined by a Transformer-style feed-forward block with residual connection and layer normalization, $\mathbf{z}_m^{(i)} = \tilde{\mathbf{z}}_m^{(i)} + \mathrm{FFN}\Big(\mathrm{LN}\Big(\tilde{\mathbf{z}}_m^{(i)}\Big)\Big)$. After M layers, we apply a final layer normalization and obtain the current-point state token $\mathbf{e}^{(i)} = \mathrm{LN}\Big(\mathbf{z}_M^{(i)}\Big)$, which represents variable i at the current point conditioned on its recent history. This produces one current-point state token for each variable before inter-variable contextualization is introduced.

Inter-Variable Layer. Temporal modeling yields a set of current-point state tokens, $\mathbf{E} = [\mathbf{e}^{(1)}; \ldots; \mathbf{e}^{(N)}]$. We then model inter-variable dependencies by applying self-attention along the variable dimension. With this ordering, TENET first constructs one current-point state token per variable and then models dependencies among variables. Concretely, $f_v(\cdot)$ is implemented as a stack of Transformer-style blocks over the variable axis, each composed of multi-head self-attention followed by a position-wise feed-forward network with residual connections and layer normalization. The final token-wise layer normalization is treated as part of this layer. The output of the inter-variable module is the contextualized variable-level state set

$$\mathbf{S} = f_v(\mathbf{E}) \in \mathbb{R}^{N \times d}.$$

Each token is updated using information from the other variables, producing a structured variable-level state set $\mathbf{S}$. This set is an intermediate representation. It preserves one current-point state token for each variable after inter-variable contextualization, but it is not yet the final decision state.

Aggregation Layer. To obtain the current-point system state used for scoring, we pool the contextualized variable-level states in $\mathbf{S}$ into $\mathbf{c} \in \mathbb{R}^h$. This aggregation design is inspired by the centralized core representation in recent multivariate time-series models [14]. We use summation so that variable-level contributions can accumulate in the system state, allowing weak deviations distributed across variables to remain visible after pooling:

$$\mathbf{c} = f_g(\mathbf{S}) = \rho\left(\sum_{i=1}^{N} \phi\left(\mathbf{s}^{(i)}\right)\right). \tag{5}$$

Here, $\phi : \mathbb{R}^d \to \mathbb{R}^h$ is a two-layer MLP with GELU activation and $\rho : \mathbb{R}^h \to \mathbb{R}^h$ is a single linear layer followed by GELU. We refer to $\mathbf{c}$ as the current-point system state. This final decision state is used as the input to our simple one-class scoring objective.

3.3 Training and Inference

We train TENET only on normal windows and use the same compatibility score at both training and inference. We deliberately adopt a simple fixed one-class objective because the practical goal of anomaly detection is not necessarily to recover the normal-state distribution as faithfully as possible, but to learn a state space that enables a useful compatibility ranking between normal and anomalous states. From this perspective, fixing the reference distribution lets the experiment focus on state construction rather than density-model complexity.

With the fixed reference distribution $p_{\mathrm{ref}}(\mathbf{c}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, we define the compatibility score as the negative log-likelihood (NLL) under this reference:

$$\mathcal{A}(\mathbf{X}) = -\log p_{\mathrm{ref}}(f_\Theta(\mathbf{X})) = \frac{1}{2}\left\|f_\Theta(\mathbf{X})\right\|_2^2 + C, \tag{6}$$

where C is constant with respect to $\mathbf{X}$. Training minimizes the average score over normal windows, encouraging TENET to place normal current-point states near the origin under the same score used at inference time. At inference time, for each time step t we form the window $\mathbf{X}_t = \mathbf{x}_{t-L+1:t}$ and assign the anomaly score of the current point $\mathbf{x}_t$ by $\mathcal{A}(\mathbf{X}_t)$. The fixed NLL objective has a constant-map global optimum, so it is important to verify whether a detector preserves anomaly-ranking information without collapse over the finite training duration used in our experiments. Additional discussion is provided in Appendix A, and an extended-training stability analysis is provided in Appendix B. Appendix J.2 compares fixed and learnable Gaussian references.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate our method on four multivariate time series anomaly detection benchmarks: **PSM** [1] (server monitoring, 25 variables), **SMD** [42] (server machine, 38 variables), **SWaT** [27] (water treatment, 51 variables), and **SWAN** [2] (solar activity, 38 variables). Dataset-selection rationale and statistics are provided in Appendix D.

Table 1: Multivariate time series anomaly detection performance on real-world datasets. All metrics are reported in percent and averaged over five random seeds. **Bold** indicates the best score.

Dataset	PSM (25 vars)				SMD (38 vars)				SWaT (51 vars)				SWAN (38 vars)			
Metric	A-ROC	A-PR	V-ROC	V-PR	A-ROC	A-PR	V-ROC	V-PR	A-ROC	A-PR	V-ROC	V-PR	A-ROC	A-PR	V-ROC	V-PR
IF	54.23	33.41	56.47	37.78	56.59	15.63	56.21	15.91	36.35	10.27	41.37	10.17	48.96	38.12	52.67	62.27
PCA	65.89	47.32	65.59	47.53	77.11	60.86	76.95	62.14	81.85	72.64	81.77	48.17	72.89	63.16	64.46	72.05
OCSVM	69.29	50.83	69.12	50.97	76.87	61.71	76.52	61.42	81.76	71.80	68.54	42.38	71.36	60.75	63.99	70.90
DAGMM	54.91	32.55	54.80	32.05	78.25	59.90	77.73	19.91	61.96	16.47	36.61	9.67	67.18	55.59	60.08	64.16
THOC	64.17	43.56	64.03	44.19	79.80	62.64	71.41	66.45	55.58	24.19	55.32	27.12	73.11	62.59	58.96	73.52
DC	49.86	27.78	50.11	27.89	50.31	10.15	49.97	10.05	49.89	12.14	50.00	11.92	49.74	32.54	50.00	55.05
CAROTS	75.48	53.03	66.05	47.41	83.26	57.46	80.54	54.06	83.10	73.28	64.27	48.31	67.60	56.60	**76.00**	69.70
Omni	63.95	45.00	61.23	52.49	**95.24**	64.35	70.26	54.99	82.15	72.73	81.98	64.42	75.63	63.32	68.36	74.43
A.T	38.35	24.28	50.96	49.76	48.44	10.73	50.24	33.60	42.58	11.93	50.72	17.00	41.69	31.27	50.14	63.13
PatchTST	58.68	37.91	50.62	50.31	66.44	23.83	50.87	30.51	24.36	8.40	49.84	11.92	55.63	47.53	50.98	75.11
TimesNet	59.09	38.95	50.67	49.86	65.18	23.37	51.22	31.05	28.97	10.71	20.80	51.20	70.85	60.71	50.48	75.41
iTrans	59.22	38.27	51.23	44.54	67.03	24.49	50.91	28.38	24.57	8.33	49.90	12.05	52.10	42.11	53.27	69.31
Modern	59.21	38.59	50.68	50.55	65.90	24.09	50.45	41.73	24.50	9.00	50.15	13.05	53.48	46.84	52.62	74.46
SARAD	62.86	41.67	57.63	38.64	93.95	59.73	53.65	31.77	85.40	64.77	86.30	62.72	71.87	60.44	56.93	77.37
CATCH	64.75	43.40	57.00	45.95	67.26	25.15	50.79	33.02	33.31	13.39	51.89	18.70	50.47	43.98	53.48	73.01
CrossAD	61.79	41.30	50.61	50.67	87.53	65.74	50.92	46.09	55.92	14.28	50.32	13.88	61.88	52.51	53.64	74.78
TTM	53.99	33.83	51.29	45.30	59.64	21.41	50.30	52.28	26.68	11.55	51.20	15.58	64.76	52.77	53.04	74.34
Timer	55.60	35.93	51.38	46.95	58.64	21.51	50.28	35.22	28.62	11.91	51.58	15.76	62.27	51.44	52.56	73.36
Chronos	56.39	36.89	51.36	52.58	58.02	15.82	50.16	13.68	79.86	38.17	78.27	34.06	52.86	45.94	53.06	72.72
GPT4TS	57.79	37.86	50.29	56.14	61.42	24.21	50.80	43.49	24.27	8.53	50.21	25.69	58.97	48.75	50.69	75.09
TENET	**86.14**	**74.27**	**85.82**	**74.21**	90.04	**74.78**	**89.50**	**74.41**	**87.78**	**79.63**	**87.67**	**79.45**	**88.36**	**84.07**	72.98	**80.97**

Evaluation Metrics. Following the evaluation practices recommended in recent studies [17, 21, 32, 37, 46], we report performance using a comprehensive set of evaluation metrics. Specifically, we report point-wise ranking metrics AUC-ROC (A-ROC) [11], AUC-PR (A-PR) [8], as well as range-aware, tolerance-aware metrics VUS-ROC (V-ROC) and VUS-PR (V-PR) [30]. Detailed definitions and evaluation protocols are provided in Appendix E.

Baselines and Encoders. The main benchmark comparison includes a broad set of MTSAD baselines spanning classical anomaly detection methods, latent reconstruction, representation learning, and forecasting-based approaches. Specifically, Table 1 includes PCA [41], OCSVM [38], Isolation Forest (IF) [20], DAGMM [56], OmniAnomaly (Omni) [42], Anomaly Transformer (A.T) [50], PatchTST [28], TimesNet [47], iTransformer (iTrans) [22], ModernTCN (Modern) [25], SARAD [7], CATCH [49], DCdetector (DC) [51], CAROTS [16], THOC [40], TTM [9], Timer [23], Chronos [3], and CrossAD [18]. For the encoder–objective comparison in Table 2, we compare TENET with five alternative sequence encoders: LSTM [26], Dilated RNN [40], Mamba [13], ModernTCN [25], and GPT4TS [55]. Each encoder produces a same-dimensional pooled state for the current point, allowing us to isolate the effect of the encoder under matched one-class objectives.

Implementation Details For TENET, we construct model inputs as fixed-length sliding windows with length $L = 20$. The variable-token dimension is set to $d = 256$, the pooled state dimension is set to $h = 256$, and the temporal embedding dimension is set to $d_t = 16$. The temporal layer f_t is implemented with a depth of $M = 2$ Temporal blocks and the inter-variable layer f_v is implemented with depth 2. All results in Table 1 are our reruns using a TAB-based pipeline, not public leaderboard values. Methods share data loading, normalization, and evaluation, while baselines retain their recommended hyperparameters and training schedules. Differences from TAB's public protocol are detailed in Appendix F.1. For TENET, we use a batch size of 1024 and train for 10 epochs on all four main benchmark datasets. TENET is optimized with AdamW [24]. For PSM, SWaT, and SWAN, we use learning rate 1×10^{-6} and for SMD, we use learning rate 5×10^{-5}.

4.2 Main Results

Benchmark Results. Table 1 summarizes the quantitative results on four benchmark datasets. Across all four datasets, TENET with the fixed NLL-based compatibility score achieves strong performance on most of the reported metrics. This result is notable because it shows that a time-point-centric state constructor paired with a simple fixed one-class objective can outperform detectors built around reconstruction or forecasting objectives. In particular, TENET consistently surpasses strong

Table 2: Encoder and one-class objective comparison on PSM. All metrics are reported in percent and averaged over five random seeds. **Bold** indicates the best score.

Objective	NLL				DeepSVDD				THOC				DAGMM			
Metric	A-ROC	A-PR	V-ROC	V-PR	A-ROC	A-PR	V-ROC	V-PR	A-ROC	A-PR	V-ROC	V-PR	A-ROC	A-PR	V-ROC	V-PR
LSTM	72.08	52.97	71.92	53.09	72.13	53.68	71.96	53.78	66.70	45.62	66.57	45.81	69.96	48.97	67.40	48.64
Dilated RNN	66.02	46.83	65.89	47.02	72.90	53.30	72.70	53.39	60.10	40.33	60.04	40.57	65.83	44.88	65.59	45.06
Mamba	64.98	40.87	64.89	41.10	70.61	46.75	70.47	46.92	66.51	40.53	66.41	40.80	71.15	48.26	70.87	48.36
ModernTCN	74.39	51.69	63.85	49.53	74.23	51.55	63.92	49.45	78.86	54.22	78.80	55.00	73.75	47.78	60.59	44.59
GPT4TS	74.67	45.58	61.04	42.00	66.47	40.99	60.57	41.44	73.27	43.21	60.03	39.69	44.91	31.85	44.94	32.34
TENET	**86.14**	**74.27**	**85.82**	**74.21**	**82.65**	**66.79**	**82.36**	**66.82**	**80.95**	**65.36**	**80.67**	**65.41**	**74.52**	**56.92**	**74.00**	**57.05**

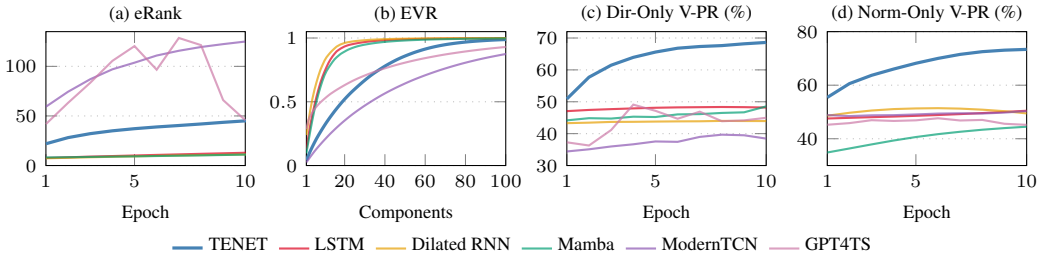

Figure 2: Representation diagnostics on PSM across TENET and alternative encoders. (a) Effective rank across epochs, where higher values indicate a more distributed representation. (b) Cumulative explained variance ratio at the final epoch, where a slower rise indicates weaker spectral concentration. (c) V-PR using only the directional component of the state vector, where higher values indicate stronger anomaly-relevant angular structure. (d) V-PR using only the radial component of the state vector, where higher values indicate stronger anomaly-relevant radial structure.

reconstruction-based baselines such as CrossAD, forecasting-based baselines such as TTM, and latent one-class MTSAD baselines such as DAGMM and THOC. On PSM, TENET improves V-PR by +21.72%pt over OmniAnomaly and +42.16%pt over DAGMM. Compared with THOC, TENET improves V-PR by +30.02%pt on PSM and +52.33%pt on SWaT. The gains on the range-aware VUS metrics are particularly notable in practice, because they indicate that TENET maintains elevated anomaly scores over anomalous segments rather than responding only at isolated timestamps. Overall, the benchmark results indicate that one-class MTSAD can outperform observation-space discrepancy scoring when paired with time-point-centric representation modeling.

One-Class Objective Generality. To examine whether TENET's advantage is specific to the default fixed NLL-based compatibility score, we construct a full encoder–objective matrix on PSM (Table 2). We compare six point-wise encoders under four representative one-class objectives, all applied to the same-dimensional pooled state $\mathbf{c} \in \mathbb{R}^h$ for the current point. This setup separates the effects of encoder design from those of objective choice.

Table 2 shows that TENET ranks first under every objective and every metric. Although the strongest non-TENET encoder varies across objectives, TENET remains consistently superior, with especially large margins under the fixed standard normal reference NLL and THOC. This result shows that TENET's advantage is not explained merely by producing a current-point state. Rather, under the same scoring setup, the key factor is how the current-point system state is constructed. These results therefore support one of our core claims: in one-class MTSAD, constructing an appropriate current-point system state is a key factor behind strong detection performance. Objective implementation details and additional encoder–objective results are provided in Appendices I and J.1, respectively.

4.3 Representation Geometry Analysis

A useful one-class state space must resolve a central tension. One-class learning puts pressure on normal states to contract toward a reference such as the origin or a hypersphere center, but when this contraction becomes too strong, the representation can collapse and lose information needed to distinguish anomalies. Figure 2 examines this tension under the same NLL objective on PSM. We use four complementary diagnostics. Effective rank [33] measures how many latent directions are substantially used, while cumulative explained variance ratio (EVR) measures how quickly variance

Table 3: Architectural variation of the temporal layer within TENET on four benchmark datasets. All metrics are reported in percent and averaged over five random seeds. **Bold** indicates the best score.

Dataset	PSM (25 vars)				SMD (38 vars)				SWaT (51 vars)				SWAN (38 vars)			
Metric	**A-ROC**	**A-PR**	**V-ROC**	**V-PR**	**A-ROC**	**A-PR**	**V-ROC**	**V-PR**	**A-ROC**	**A-PR**	**V-ROC**	**V-PR**	**A-ROC**	**A-PR**	**V-ROC**	**V-PR**
LSTM	66.58	49.38	66.47	49.57	84.70	69.75	84.07	69.84	82.40	72.47	82.31	68.80	63.54	54.41	59.46	67.99
Dilated RNN	66.64	49.36	66.52	49.53	84.91	69.77	84.16	69.77	82.40	72.59	82.30	68.76	64.19	55.41	59.47	68.10
Mamba	76.42	60.64	76.24	60.72	88.97	68.62	88.42	68.35	82.74	62.21	82.66	61.69	80.08	68.76	68.77	74.58
TENET	**86.14**	**74.27**	**85.82**	**74.21**	**90.04**	**74.78**	**89.50**	**74.41**	**87.78**	**79.63**	**87.67**	**79.45**	**88.36**	**84.07**	**72.98**	**80.97**

Table 4: Component ablation study of TENET on four benchmark datasets. All metrics are reported in percent and averaged over five random seeds. **Bold** indicates the best score.

Components				PSM (25 vars)				SMD (38 vars)				SWaT (51 vars)				SWAN (38 vars)			
Gating Mech.	**Lag Embed.**	**Per-Var. Encoder**	**Cross-Var. Attn.**	**A-ROC**	**A-PR**	**V-ROC**	**V-PR**	**A-ROC**	**A-PR**	**V-ROC**	**V-PR**	**A-ROC**	**A-PR**	**V-ROC**	**V-PR**	**A-ROC**	**A-PR**	**V-ROC**	**V-PR**
✗	✓	✓	✓	82.07	66.90	81.81	66.91	90.27	73.66	89.71	73.29	86.51	77.07	86.42	76.60	78.72	77.44	64.44	76.06
✓	✗	✓	✓	82.30	67.55	82.01	67.56	88.83	72.90	88.31	72.60	86.08	74.71	85.98	74.39	87.22	81.97	72.10	79.97
✓	✓	✗	✓	54.29	32.18	54.05	32.51	70.49	30.22	70.29	30.49	81.88	72.34	81.84	70.17	83.33	77.01	68.97	76.91
✓	✓	✓	✗	84.70	72.47	84.36	72.41	89.76	74.49	89.22	74.03	87.60	79.09	87.49	78.75	88.18	83.93	72.91	80.96
✓	✓	✓	✓	**86.14**	**74.27**	**85.82**	**74.21**	**90.04**	**74.78**	**89.50**	**74.41**	**87.78**	**79.63**	**87.67**	**79.45**	**88.36**	**84.07**	**72.98**	**80.97**

concentrates in the leading principal directions. Direction-only and norm-only probes recompute V-PR after scoring only angular deviation or radial magnitude, respectively, to test which parts of the state retain anomaly-discriminative information.

The first two panels show that TENET avoids the low-rank collapse observed in several alternative encoders. In Figure 2(a), TENET stabilizes near an effective rank of $\sim$69, whereas several alternatives contract to much lower-rank representations at the final epoch. Figure 2(b) shows the same pattern spectrally: TENET spreads variance much more gradually, reaching only 31% cumulative EVR in the top 10 components, while lower-rank alternatives already concentrate most variance in a small number of directions. However, broad geometry alone is not the target. ModernTCN and GPT4TS also occupy relatively broad subspaces, yet their anomaly detection performance remains weaker than TENET. The lesson from Figure 2 is therefore not that a wider representation is automatically better, but that the representation must be distributed in a way that remains aligned with anomaly scoring. The last two panels clarify what this alignment means. In Figure 2(c), TENET reaches 68.6% V-PR using only directional information, indicating that anomaly-relevant angular structure is retained in the learned state. In Figure 2(d), TENET also remains strongest under a norm-only score, showing that radial information is organized as well. Although the deployed fixed Gaussian NLL is radial, these probes suggest that TENET does not merely learn a trivial norm separation. Instead, its encoder organizes both radial and angular structure so that anomaly-relevant variation remains separable in the learned state space. Detailed definitions of diagnostics are provided in Appendix K.

4.4 Temporal Layer Variation

Table 3 reports results when the temporal layer in TENET is replaced with alternative sequence encoders while the rest of the model is kept fixed. Specifically, we replace f_t with LSTM, Dilated RNN, and Mamba, while keeping f_v, f_g, and the scoring head unchanged. Under this controlled substitution, TENET consistently outperforms all alternatives across the four datasets. Among the replacement architectures, Mamba is generally the strongest baseline, whereas LSTM and Dilated RNN show the largest overall degradation. These trends suggest that generic recurrent and state-space encoders can capture useful local dynamics, but still fall short of TENET's current-point-conditioned temporal retrieval. Overall, these results show that TENET's temporal layer is not an interchangeable backbone module. For point-wise anomaly detection, temporal context should be retrieved with respect to the current point, not passively compressed into a generic sequence summary.

4.5 Component Ablation

Table 4 summarizes ablations of the core components in TENET. Removing the gating mechanism in f_t degrades overall performance across the datasets, especially on the range-aware metrics. In this variant, the variable token is formed solely from the attention-retrieved temporal context, without an explicit input-dependent gate between the anchor state and retrieved history. The resulting drop

Table 5: Computational scalability of TENET on synthetic data with $d = 256$. Latency is reported in milliseconds(ms) and memory in gigabytes(GB).

Window	L=20				L=50				L=100				L=150			
Phase	Inference		Training		Inference		Training		Inference		Training		Inference		Training	
Metric	Lat.	Mem.	Lat.	Mem.	Lat.	Mem.	Lat.	Mem.	Lat.	Mem.	Lat.	Mem.	Lat.	Mem.	Lat.	Mem.
N=25	8.30	0.20	39.30	1.01	8.30	0.57	39.90	1.48	8.40	0.58	40.10	2.25	8.40	0.58	50.00	3.02
N=100	32.70	1.48	186.70	3.97	32.80	2.18	189.70	5.82	32.90	2.18	188.70	8.93	32.80	2.19	216.40	11.97
N=400	143.20	5.82	883.90	16.73	144.40	8.60	906.30	24.14	146.70	8.62	910.10	36.55	148.40	8.64	1020.20	48.75

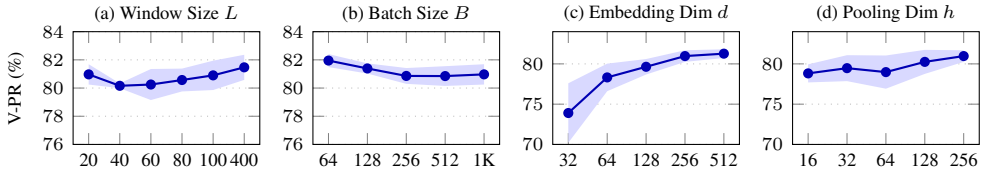

Figure 3: Hyperparameter sensitivity on SWAN. Lines show V-PR (%) averaged over five seeds and shaded regions indicate ± 1 standard deviation. X-axis settings are equally spaced.

(e.g., -7.3%pt V-PR on PSM and -2.9%pt on SWaT) indicates that temporal evidence should not be incorporated uniformly, but should instead be modulated in a variable- and input-dependent manner.

Replacing the anchor-centered lag embedding with standard sinusoidal positional encoding also reduces performance, suggesting that temporal order is more effective when encoded relative to the current point rather than by absolute position in the window. The largest degradation occurs when independent per-variable encoders are replaced by a single shared encoder. On PSM, this change lowers V-PR by 41.7%pt, which is by far the largest loss among all ablations. This supports the importance of variable-wise temporal modeling in TENET. Cross-variable attention has a more modest effect, but it provides additional refinement by composing the variable-level states. These ablation results support our design choice of TENET.

4.6 Computational Scalability

For fixed hidden dimension, TENET has two main per-window computational terms: $\mathcal{O}(NL)$ from the temporal layer and $\mathcal{O}(N^2)$ from the inter-variable layer. Table 5 reports measured latency and peak memory for inference (batch size 1) and training (batch size 64) on synthetic data over an $N \times L$ grid. The measurements show that inference latency is primarily sensitive to the number of variables and changes only weakly with the window length in the measured range. Across the tested grid, the observed growth with N is closer to linear than quadratic, indicating that the inter-variable attention term is not the sole runtime bottleneck at these problem sizes. Training memory includes parameters, gradients, optimizer states, and activations retained for backpropagation, and grows with both N and L. Training latency is mainly sensitive to N with moderate additional dependence on L. To examine memory use when both N and L are large, we profile $N = L = 200$ with $d = 256$, batch size 1, and FP32 precision. Parameter storage is 1,394.36 MB, inference adds 5.27 MB over the pre-forward memory baseline, and peak memory is 1,409.82 MB. Thus, parameters account for most inference memory in this setting. System configuration details are provided in Appendix G. A cross-model cost comparison is provided in Appendix J.3.

4.7 Hyperparameter Sensitivity

Figure 3 illustrates the sensitivity of TENET to four hyperparameters on SWAN. Overall, performance remains stable across a broad range of settings, indicating that TENET is not highly sensitive to moderate hyperparameter changes. Window-size results remain stable from $L = 20$ to $L = 400$, while batch-size changes have little influence on performance. For $L = 400$, we use micro-batches of 128 with eight-step gradient accumulation for an effective batch size of 1024. The embedding dimension d and pooling dimension h show clearer effects than window size or batch size, but in both cases performance stabilizes once a moderate capacity is reached. This suggests that TENET is robust to hyperparameter choice and does not require delicate tuning to achieve strong performance. Additional sensitivity tests under training contamination and test-time current-point perturbations are provided in Appendix L.

5 Conclusion

We presented TENET, a current-point-centric encoder for multivariate time series anomaly detection. TENET is motivated by the view that one-class MTSAD should not only ask how normality is scored, but also what representation is being scored. Since MTSAD decisions are made for specific time points, TENET treats the input window not as the object to reconstruct, forecast, summarize, or score, but as decision-relevant context for constructing the current-point system state.

Across benchmark datasets, TENET achieves strong detection performance with a simple fixed NLL-based compatibility score. Controlled comparisons further suggest that its advantage comes from decision-aligned representation construction rather than from a particular scoring rule. Our findings suggest that one-class MTSAD benefits from interpreting the decision target as a time point and designing time-point encoders that construct the state to be scored. Additional discussion of limitations and future directions is provided in the appendix N.

References

[1] Ahmed Abdulaal, Zhuanghua Liu, and Tomer Lancewicki. Practical approach to asynchronous multivariate time series anomaly detection and localization. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2021.

[2] Rafal Angryk, Petrus Martens, Berkay Aydin, Dustin Kempton, Sushant Mahajan, Sunitha Basodi, Azim Ahmadzadeh, Xumin Cai, Soukaina Filali Boubrahimi, Shah Muhammad Hamdi, Micheal Schuh, and Manolis Georgoulis. SWAN-SF. *Harvard Dataverse Dataset*, 2020.

[3] Abdul Fatir Ansari, Lorenzo Stella, Ali Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Bernie Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024.

[4] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A. Zuluaga. Do deep neural networks contribute to multivariate time series anomaly detection? *Pattern Recognition*, 2022.

[5] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[6] Dongchan Cho, Jiho Han, Keumyeong Kang, Minsang Kim, Honggyu Ryu, and Namsoon Jung. Structured temporal causality for interpretable multivariate time series anomaly detection. In *Advances in Neural Information Processing Systems*, 2025.

[7] Zhihao Dai, Ligang He, Shuang-Hua Yang, and Matthew Leeke. SARAD: Spatial association-aware anomaly detection and diagnosis for multivariate time series. In *Advances in Neural Information Processing Systems*, 2024.

[8] Jesse Davis and Mark Goadrich. The relationship between precision-recall and ROC curves. In *International Conference on Machine Learning*, 2006.

[9] Vijay Ekambaram, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam H. Nguyen, Wesley M. Gifford, Chandra Reddy, and Jayant Kalagnanam. Tiny time mixers (TTMs): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series. In *Advances in Neural Information Processing Systems*, 2024.

[10] Kamil Faber, Marcin Pietroń, Dominik Żurek, and Roberto Corizzo. xLSTMAD: A powerful xLSTM-based method for anomaly detection. In *IEEE International Conference on Data Mining*, 2025.

[11] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 2006.

[12] Mononito Goswami, Cristian Ignacio Challu, Laurent Callot, Lenon Minorics, and Andrey Kan. Unsupervised model selection for time series anomaly detection. In *International Conference on Learning Representations*, 2023.

[13] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *Conference on Language Modeling*, 2024.

[14] Lu Han, Xu-Yang Chen, Han-Jia Ye, and De-Chuan Zhan. SOFTS: Efficient multivariate time series forecasting with series-core fusion. In *Advances in Neural Information Processing Systems*, 2024.

[15] Lu Han, Han-Jia Ye, and De-Chuan Zhan. The capacity and robustness trade-off: Revisiting the channel independent strategy for multivariate time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[16] 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 *International Conference on Machine Learning*, 2025.

[17] Siwon Kim, Kukjin Choi, Hyun-Soo Choi, Byunghan Lee, and Sungroh Yoon. Towards a rigorous evaluation of time-series anomaly detection. In *AAAI Conference on Artificial Intelligence*, 2022.

[18] 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 *Advances in Neural Information Processing Systems*, 2025.

[19] Gang Li, Mingchao Ge, Jin Wan, Delong Han, Min Li, and Mingle Zhou. MemMambaAD: Memory-augmented state space model for multivariate time series anomaly detection. *Engineering Applications of Artificial Intelligence*, 2025.

[20] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *IEEE International Conference on Data Mining*, 2008.

[21] Qinghua Liu and John Paparrizos. The Elephant in the Room: Towards a reliable time-series anomaly detection benchmark. In *Advances in Neural Information Processing Systems*, 2024.

[22] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. iTransformer: Inverted transformers are effective for time series forecasting. In *International Conference on Learning Representations*, 2024.

[23] 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*, 2024.

[24] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

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

[26] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. LSTM-based encoder-decoder for multi-sensor anomaly detection. In *ICML Workshop on Anomaly Detection*, 2016.

[27] Aditya P. Mathur and Nils Ole Tippenhauer. SWaT: A water treatment testbed for research and training on ICS security. In *International Workshop on Cyber-Physical Systems for Smart Water Networks*, 2016.

[28] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A Time Series is Worth 64 Words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

[29] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton van den Hengel. Deep learning for anomaly detection: A review. *ACM Computing Surveys*, 2021.

[30] 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. *Proceedings of the VLDB Endowment*, 2022.

[31] Jinju Park and Seokho Kang. PaAno: Patch-based representation learning for time-series anomaly detection. In *International Conference on Learning Representations*, 2026.

[32] 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. *Proceedings of the VLDB Endowment*, 2025.

[33] Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *European Signal Processing Conference*, 2007.

[34] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International Conference on Machine Learning*, 2018.

[35] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Workshop on Machine Learning for Sensory Data Analysis*, 2014.

[36] M. Saquib Sarfraz, Mei-Yen Chen, Lukas Layer, Kunyu Peng, and Marios Koulakis. Position: Quo vadis, unsupervised time series anomaly detection? In *International Conference on Machine Learning*, 2024.

[37] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. Anomaly detection in time series: A comprehensive evaluation. *Proceedings of the VLDB Endowment*, 2022.

[38] Bernhard Schölkopf, Robert C. Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection. In *Advances in Neural Information Processing Systems*, 1999.

[39] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2018.

[40] Lifeng Shen, Zhuocong Li, and James Kwok. Timeseries anomaly detection using temporal hierarchical one-class network. In *Advances in Neural Information Processing Systems*, 2020.

[41] Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. A novel anomaly detection scheme based on principal component classifier. In *ICDM Workshop on Foundation and New Direction of Data Mining*, 2003.

[42] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2019.

[43] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, 2014.

[44] Shreshth Tuli, Giuliano Casale, and Nicholas R. Jennings. TranAD: Deep transformer networks for anomaly detection in multivariate time series data. *Proceedings of the VLDB Endowment*, 2022.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[46] Dennis Wagner, Tobias Michels, Florian C. F. Schulz, Arjun Nair, Maja Rudolph, and Marius Kloft. TimeSeAD: Benchmarking deep multivariate time-series anomaly detection. *Transactions on Machine Learning Research*, 2023.

[47] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. TimesNet: Temporal 2D-variation modeling for general time series analysis. In *International Conference on Learning Representations*, 2023.

[48] Renjie Wu and Eamonn J. Keogh. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[49] 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 *International Conference on Learning Representations*, 2025.

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

[51] Yiyuan Yang, Chaoli Zhang, Tian Zhou, Qingsong Wen, and Liang Sun. DCdetector: Dual attention contrastive representation learning for time series anomaly detection. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023.

[52] Tao Yin, Xiaohong Zhang, Shaochen Fu, Zhibin Zhang, Li Huang, Yiyuan Yang, Kaixiang Yang, and Meng Yan. ScatterAD: Temporal-topological scattering mechanism for time series anomaly detection. In *Advances in Neural Information Processing Systems*, 2025.

[53] Zahra Zamanzadeh Darban, Geoffrey I. Webb, Shirui Pan, Charu C. Aggarwal, and Mahsa Salehi. Deep learning for time series anomaly detection: A survey. *ACM Computing Surveys*, 2024.

[54] Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *International Conference on Learning Representations*, 2023.

[55] Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. One fits all: Power general time series analysis by pretrained LM. In *Advances in Neural Information Processing Systems*, 2023.

[56] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding Gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*, 2018.

A Design Rationale for TENET

This appendix provides an informal design-level interpretation of why TENET's current-point-centric state constructor can produce the distributed and anomaly-discriminative geometry observed in Figure 2. The fixed standard-normal NLL used in the main experiments is intentionally simple:

$$A(\mathbf{X}) = \frac{1}{2}\|f_\Theta(\mathbf{X})\|_2^2 + C.$$

We use this score as a fixed compatibility interface rather than as a new one-class objective or a calibrated density estimator. Its role is to keep the scoring geometry simple so that the empirical effect of the state constructor can be isolated. As with compact one-class objectives based on a fixed reference, the loss can admit degenerate constant solutions in principle.

The operational goal in unsupervised MTSAD is not necessarily to recover the full normal-state density, but to learn an anomaly score whose test-time ordering ranks abnormal current-point states above normal ones. Under the fixed NLL score, this ordering is induced by the radial magnitude of the learned current-point system state. The relevant empirical question is therefore whether normal-only training with this simple compatibility score produces nontrivial states whose inference-time scores separate normal and anomalous points. The benchmark results in Section 4.2 evaluate this directly with threshold-free ranking metrics, and the encoder–objective comparisons in Table 2 and Appendix J.1 test whether TENET's advantage persists beyond the default NLL score.

TENET is designed around the hypothesis that this ranking depends primarily on what state is constructed before scoring. Its temporal layer first forms one current-point token per variable rather than compressing the whole multivariate window into a single generic summary. This variable-wise temporal construction reduces early cross-variable interference and allows different variables to retain their own temporal patterns before multivariate composition. The shared-encoder ablation in Table 4 supports this interpretation, although it also changes model capacity and should therefore be viewed as evidence for, not a complete isolation of, variable-specific specialization.

The temporal block retrieves history with respect to the current point. At each block, the query is derived from the evolving current-point anchor, while keys and values come from the window context. Thus, the model can select historical information according to its relevance to the present state rather than applying a fixed window summarization rule. Anchor-centered lag embeddings express time relative to the current decision point, and the gate adaptively mixes the current anchor path with the retrieved-history path. These components are intended to preserve information that helps explain the current point while avoiding uniform accumulation of all past observations.

After variable-wise temporal construction, TENET applies cross-variable attention over the resulting current-point tokens. This ordering is important because inter-variable modeling is performed after each variable has already selected its relevant temporal context. The aggregation layer then maps the structured variable-level state set into a single current-point system state for scoring. Consequently, the fixed NLL is not applied to a raw window summary, but to a state produced by variable-wise temporal retrieval, cross-variable contextualization, and nonlinear set aggregation.

This design rationale is consistent with the representation diagnostics in Figure 2. TENET learns a state space that is less spectrally concentrated than several alternative encoders and retains anomaly-discriminative information under both radial and directional probes. The deployed NLL score uses only radial magnitude, so the directional probe is diagnostic rather than part of the detector. Its purpose is to show that the learned representation is not merely a trivial scalar norm separation. Overall, TENET's advantage should be understood as coming from decision-aligned current-point state construction rather than from making the one-class objective more complex.

B Extended-Training Stability

The main TENET results use the fixed NLL objective in Eq. (6), which encourages compact current-point states. We therefore include an extended-training stability analysis to assess the learned representation under prolonged optimization. Specifically, we train TENET on PSM for 100 epochs, which is ten times longer than the default 10-epoch protocol used in the main experiments, and log per-epoch diagnostics. Figure S1 reports eight quantities side by side: (a) test mean state norm, (b) RMS pairwise distance among pooled states, (c) effective rank of the pooled-state covariance, (d) the cumulative explained-variance ratio at the final epoch, and (e)–(h) the four detection metrics (A-ROC, A-PR, V-ROC, V-PR) on the test split. Test mean state norm and RMS pairwise distance diagnose compactness, where decreasing values reflect a more compact pooled-state distribution. Effective rank [33] and cumulative EVR diagnose whether increasing compactness is accompanied by low-rank behavior, either by reducing the number of active covariance directions or by concentrating variance in a few principal components. The detection metrics track whether the changing representation still preserves anomaly-ranking information.

Compactness and Detection Performance. As expected from a one-class compactness objective, the pooled states become increasingly compact around the origin. This is visible in the continued decrease of the mean state norm and the RMS pairwise distance among pooled states. Figure S1(a) further shows that normal and anomalous states move toward smaller norms together, but their norm gap remains visible across training. Thus, increasing compactness does not immediately erase the anomaly-ranking signal used by the fixed NLL score. All four detection metrics continue to improve after the default epoch-10 checkpoint, reach their highest values later in training, and remain strong even under much longer training. This suggests that TENET retains useful anomaly-ranking information under the fixed NLL objective, even as the pooled states become more compact.

Compactness and Representation Diversity. The representation diagnostics do not indicate simple low-rank collapse under extended training. Effective rank continues to grow, and the cumulative EVR curve shows that the remaining variance is distributed across many directions rather than concentrated in only a few components. Thus, prolonged NLL training mainly induces compactness-driven scale reduction and latent-state drift, while the learned state remains informative for anomaly ranking.

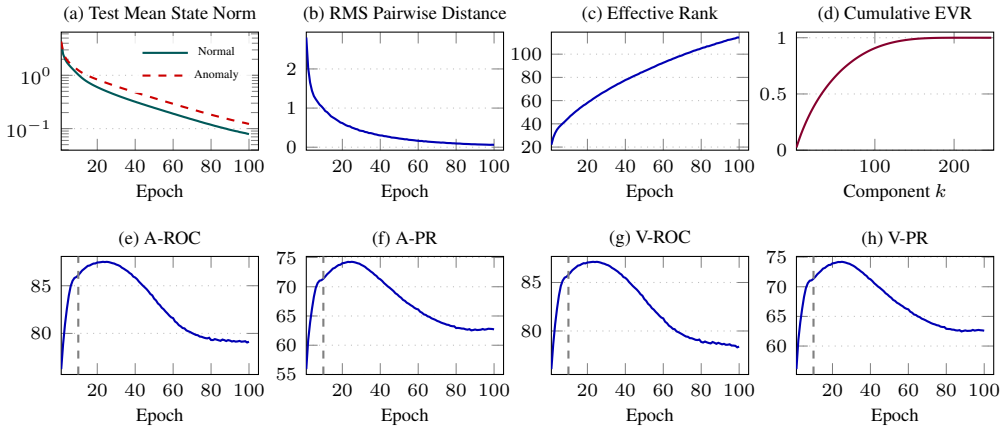

Figure S1: Extended-training stability analysis for TENET on PSM. Panels (a)–(d) show test mean state norm, RMS pairwise distance, effective rank, and cumulative EVR. Panels (e)–(h) report A-ROC, A-PR, V-ROC, and V-PR over training.

C Pseudocode

This appendix presents implementation-level pseudocode for TENET. The notation follows the PyTorch execution path used in our experiments. A dataloader returns a minibatch $\mathbf{X}_{\text{raw}} \in \mathbb{R}^{B\times L\times N}$, where B is the batch size, L is the window length, and N is the number of variables. The batch is transposed to $\mathbf{X} \in \mathbb{R}^{B\times N\times L}$ so that each variable can be encoded independently.

Algorithm 1 TENET pseudocode

Input: Minibatch $\mathbf{X} \in \mathbb{R}^{B\times N\times L}$, variable-specific temporal encoders $\{f_t^{(i)}\}_{i=1}^{N}$, inter-variable layer f_v, aggregation functions ϕ, ρ
Output: Current-point system state $\mathbf{c}$ and anomaly score vector $\mathbf{a}$
Initialize token list $\mathcal{S} \leftarrow [\,]$
for $i = 1$ to N **do**
  Extract variable window $\mathbf{X}^{(i)} \leftarrow \mathbf{X}_{[:,i,:]} \in \mathbb{R}^{B\times L}$
  Reshape $\mathbf{X}^{(i)}$ to $B \times L \times 1$
  Project scalar values to embeddings $\mathbf{E}^{(i)} \leftarrow \mathrm{Linear}(\mathbf{X}^{(i)}) \in \mathbb{R}^{B\times L\times d}$
  Initialize the current anchor $\mathbf{z}^{(i)} \leftarrow \mathbf{E}^{(i)}_{[:,L,:]} \in \mathbb{R}^{B\times d}$
  Build current-relative lag indices $[L-1, \ldots, 0]$ and look up lag embeddings $\mathbf{R} \in \mathbb{R}^{B\times L\times d_t}$
  Form key-value tokens $\mathbf{U}^{(i)} \leftarrow [\mathbf{E}^{(i)}, \mathbf{R}] \in \mathbb{R}^{B\times L\times (d+d_t)}$
  for Temporal block $m = 1$ to M **do**
    $\mathbf{q} \leftarrow \mathrm{LN}_q(\mathbf{z}^{(i)})\mathbf{W}_Q \in \mathbb{R}^{B\times 1\times d}$
    $\mathbf{K} \leftarrow \mathrm{LN}_{kv}(\mathbf{U}^{(i)})\mathbf{W}_K, \quad \mathbf{V} \leftarrow \mathrm{LN}_{kv}(\mathbf{U}^{(i)})\mathbf{W}_V$
    Split $\mathbf{q}, \mathbf{K}, \mathbf{V}$ into H heads
    $\boldsymbol{\alpha} \leftarrow \mathrm{softmax}(\mathbf{q}\mathbf{K}^\top/\sqrt{d/H})$
    $\mathbf{h} \leftarrow \mathrm{merge}(\boldsymbol{\alpha}\mathbf{V})\mathbf{W}_O \in \mathbb{R}^{B\times 1\times d}$
    $\mathbf{g} \leftarrow \sigma(\mathrm{Linear}(\mathbf{q}))$
    $\tilde{\mathbf{z}} \leftarrow \mathbf{g} \odot \mathbf{q} + (1-\mathbf{g}) \odot \mathbf{h}$
    $\mathbf{z}^{(i)} \leftarrow \mathrm{squeeze}(\tilde{\mathbf{z}}) + \mathrm{FFN}(\mathrm{LN}(\mathrm{squeeze}(\tilde{\mathbf{z}})))$
  end for
  Append $\mathrm{LN}(\mathbf{z}^{(i)})$ to $\mathcal{S}$
end for
Stack variable tokens $\mathbf{S}_0 \leftarrow \mathrm{stack}(\mathcal{S}, \mathrm{dim} = 1) \in \mathbb{R}^{B\times N\times d}$
Contextualize and normalize variables $\mathbf{S} \leftarrow f_v(\mathbf{S}_0) \in \mathbb{R}^{B\times N\times d}$
Pool variable states $\mathbf{c} \leftarrow \rho(\sum_{i=1}^{N} \phi(\mathbf{s}^{(i)})) \in \mathbb{R}^{B\times h}$
Compute fixed standard-normal NLL $a_b \leftarrow \frac{1}{2}\sum_{j=1}^{h}\left(c_{b,j}^2 + \log(2\pi)\right)$ for each sample b
return $\mathbf{c}, \mathbf{a}$

Training and inference use the same forward path in Algorithm 1. During training, Algorithm 1 is applied to normal windows and the model minimizes $\mathcal{L}_{\text{NLL}} = B^{-1}\sum_{b=1}^{B} a_b$ with AdamW. During inference, the returned NLL value a_b is used directly as the raw anomaly score for the current time point. The raw scores are evaluated with AUC-ROC, AUC-PR, VUS-ROC, and VUS-PR without score smoothing, point adjustment, or reconstruction discrepancy.

D Datasets

Table S1: Statistics of the benchmark datasets used in the experiments. Anomaly count, anomaly ratio, and segment lengths are computed from the test labels, where count denotes the number of contiguous anomalous segments.

Dataset	PSM	SMD	SWaT	SWAN
Features	25	38	51	38
Set size	Train: 132,481 Test: 87,841	Train: 23,697 Test: 23,697	Train: 496,800 Test: 449,919	Train: 60,000 Test: 60,000
Anomalies	Count: 71 Ratio: 27.73%	Count: 13 Ratio: 10.12%	Count: 34 Ratio: 12.02%	Count: 6,148 Ratio: 32.60%
Segment Lengths	Min: 1 Med: 5 Max: 8,861	Min: 3 Med: 26 Max: 1,215	Min: 101 Med: 447 Max: 35,900	Min: 1 Med: 1 Max: 11,442
Sampling Period	1 min	1 min	1 sec	12 min

D.1 Dataset Selection

For the main benchmark, we considered real-world datasets with at least 20 variables as a practical scope for studying variable-wise temporal encoding and cross-variable modeling. We selected complementary anomaly durations and frequencies rather than aiming to cover every domain. PSM includes short transients and sustained anomalies, with labeled segment lengths from 1 to 8,861 points. SMD (`machine-1-7`) tests detection with a lower anomaly ratio of 10.12%, while SWaT includes sustained process anomalies lasting up to 35,900 points. SWAN tests point-like events, with 6,148 labeled segments and a median length of one point. We use the original releases to preserve their train–test splits and sequence boundaries, rather than selecting these four datasets from TAB's curated collection.

D.2 Dataset Statistics

Table S1 summarizes the benchmark datasets used in our experiments. We report not only the dimensionality and split sizes, but also the segment-level characteristics of anomalies. Here, count denotes the number of contiguous anomalous segments in the test labels, and segment lengths are computed over these contiguous segments (minimum/median/maximum). These datasets differ substantially in terms of dimensionality, time series length, anomaly density, and the temporal structure of anomalous events.

PSM [1] contains relatively few but potentially long anomaly segments, resulting in a high anomaly ratio despite a limited number of events. **SMD** [42] (`machine-1-7`) has a moderate number of short-to-medium anomaly segments with $\sim$10% density. **SWaT** [27] exhibits a wide range of anomaly durations, spanning from short abnormal intervals to extremely long segments, which stresses robustness under heterogeneous attack effects. **SWAN** [2] has the densest anomaly structure among our benchmarks, containing thousands of very short anomaly segments intermixed with longer ones, which tests stability under heavily contaminated streams.

D.3 Dataset Descriptions

PSM (Pooled Server Metrics). PSM [1] is a multivariate server-metrics dataset released in the context of practical anomaly detection research, containing 25 variables with minute-level sampling. Its test split includes both short spikes and prolonged deviations, creating a challenging mixture of dense and long-duration anomalies.

SMD (Server Machine Dataset). SMD [42] consists of 28 independent machine subsets, each with 38 variables. We report results on `machine-1-7`, following recent evaluation practice that avoids averaging over highly heterogeneous machines. This machine contains 23,697 training samples and 23,697 test samples at 1-minute sampling.

SWaT (Secure Water Treatment). SWaT [27] is an industrial control system (ICS) dataset collected from a realistic water treatment testbed, with 51 tags (sensors and actuators) sampled every second. It was recorded over 11 days of continuous operation, where the first 7 days correspond to normal operation and the remaining days include cyber/physical attack interventions. The test split contains attack periods that induce diverse anomaly durations and multivariate dependency shifts.

SWAN (Space Weather ANalytics). SWAN [2] is a solar-activity monitoring dataset collected from NASA SDO/HMI vector magnetograms, with 38 numerical features describing physical properties of active regions (e.g., total unsigned flux, magnetic free energy, helicity). Samples are recorded every 12 minutes, and anomalies correspond to pre-flare and flare states of solar active regions. The train and test splits each contain 60,000 time steps, and anomalies appear as densely packed short segments mixed with longer event windows.

D.4 Dataset Sources and Access

We use publicly available benchmarks under their respective terms of use:

- **PSM (Pooled Server Metrics)**[2]: Released as part of the `eBay/RANSynCoders` repository. The repository documentation specifies that the sample data are released under Creative Commons Attribution 4.0 (see `data/LICENSE`), while the non-data repository content is released under BSD-3 (see `LICENSE`).
- **SMD (Server Machine Dataset)**[3]: Distributed via the `NetManAIOps/OmniAnomaly` repository, which states that the 28 subsets should be trained and tested separately, and provides an MIT license for the repository.
- **SWaT (Secure Water Treatment)**[4]: Provided by iTrust, SUTD. Access typically requires a request via the dataset portal, and users must comply with the stated terms (e.g., attribution requirements and restrictions on redistribution).
- **SWAN (Space Weather ANalytics)**[5]: Released on Harvard Dataverse under an open license for academic research on solar activity and space-weather forecasting.

[2] https://github.com/eBay/RANSynCoders

[3] https://github.com/NetManAIOps/OmniAnomaly

[4] https://itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info/

[5] https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/EBCFKM

E Evaluation Metrics

Table S2: Summary of the evaluation metrics used in the experiments.

Metric	AUC-ROC	AUC-PR	VUS-ROC	VUS-PR
Granularity	Point-wise	Point-wise	Point+Time	Point+Time
Threshold-Free	Yes	Yes	Yes	Yes
Temporal-Aware	No	No	Yes	Yes
Delay-Tolerant	No	No	Yes	Yes
Imbalance-Robust	No	Yes	Yes	Yes

We evaluate detection performance using four threshold-free metrics, **AUC-ROC** [11], **AUC-PR** [8], **VUS-ROC**, and **VUS-PR** [30]. These metrics assess the ranking quality of anomaly scores without committing to a fixed operating threshold. Since time-series anomalies are often sparse and temporally structured, we report both ROC and PR metrics and additionally include range-aware VUS metrics to account for temporal tolerance around anomalous intervals.

Unless stated otherwise, we directly use the raw anomaly scores produced by the model, without any score post-processing. We focus on threshold-free metrics because evaluating a detector with a test-set optimal threshold is not operationally meaningful. It implicitly assumes access to future test labels and overestimates deploy-time performance. Using a pre-selected threshold is also method-dependent and typically requires strong assumptions (e.g., consistent train/test score distributions and stable score distributions across deployment data), which may not hold in real systems.

E.1 AUC-ROC and AUC-PR

Let $\{s_t\}_{t=1}^T$ be the anomaly score sequence and $\{y_t\}_{t=1}^T$ the binary labels. For a threshold τ, predictions are $\hat{y}_t(\tau) = \mathbb{I}[s_t \geq \tau]$, yielding $TP(\tau)$, $FP(\tau)$, $TN(\tau)$, and $FN(\tau)$. The true positive rate (TPR) and false positive rate (FPR) are

$$\begin{aligned} \mathrm{TPR}(\tau) &= \frac{TP(\tau)}{TP(\tau) + FN(\tau)}, \\ \mathrm{FPR}(\tau) &= \frac{FP(\tau)}{FP(\tau) + TN(\tau)}. \end{aligned} \tag{S1}$$

AUC-ROC is defined as the area under the ROC curve parameterized by τ:

$$\text{AUC-ROC} = \int \mathrm{TPR}(\tau)\, d\,\mathrm{FPR}(\tau). \tag{S2}$$

In practice, these AUC metrics are computed by sweeping τ over all distinct score values (i.e., all possible decision thresholds), as implemented in standard toolkits.

Similarly, precision and recall at threshold τ are

$$\begin{aligned} \mathrm{Precision}(\tau) &= \frac{TP(\tau)}{TP(\tau) + FP(\tau)}, \\ \mathrm{Recall}(\tau) &= \frac{TP(\tau)}{TP(\tau) + FN(\tau)}. \end{aligned} \tag{S3}$$

AUC-PR is the area under the precision–recall curve:

$$\text{AUC-PR} = \int \mathrm{Precision}(\tau)\, d\,\mathrm{Recall}(\tau). \tag{S4}$$

In multivariate time-series anomaly detection, anomalies are typically rare. As a result, AUC-ROC can be overly optimistic because FPR is diluted by the large number of true negatives. AUC-PR is often more informative in this regime since it focuses on the positive class and better reflects the practical trade-off between false alarms and missed anomalies. We therefore report both metrics,

and interpret AUC-PR as the more reliable indicator under severe imbalance. Both AUC-ROC and AUC-PR are temporally agnostic. They do not measure detection continuity, localization lag, or tolerance around anomalous ranges. To explicitly account for temporal tolerance while remaining threshold-free, we additionally report VUS metrics.

E.2 VUS-ROC and VUS-PR

Our implementation of Volume Under the Surface (VUS) metrics follows a range-aware threshold-free evaluation scheme. Let $\Omega = \{0, 1, \ldots, 10\}$ be the set of temporal tolerance windows used in the main benchmark. For each $\omega \in \Omega$, we construct a softened label sequence $\tilde{y}_t^{(\omega)} \in [0, 1]$ by extending each anomalous interval by $\lfloor \omega/2 \rfloor$ time steps on both sides and assigning distance-dependent fractional credit near the interval boundaries. Given anomaly scores $\{s_t\}_{t=1}^{T}$ and threshold τ, predictions are $\hat{y}_t(\tau) = \mathbb{I}[s_t \geq \tau]$. For each ω and τ, we compute soft true-positive, false-positive, false-negative, and true-negative counts as

$$\begin{aligned} \mathrm{TP}_\omega(\tau) &= \sum_t \tilde{y}_t^{(\omega)} \hat{y}_t(\tau), & \mathrm{FP}_\omega(\tau) &= \sum_t \left(1 - \tilde{y}_t^{(\omega)}\right) \hat{y}_t(\tau), \\ \mathrm{FN}_\omega(\tau) &= \sum_t \tilde{y}_t^{(\omega)} - \mathrm{TP}_\omega(\tau), & \mathrm{TN}_\omega(\tau) &= \sum_t \left(1 - \tilde{y}_t^{(\omega)}\right) - \mathrm{FP}_\omega(\tau). \end{aligned} \tag{S5}$$

These quantities define tolerance-dependent ROC and PR surfaces through $\mathrm{TPR}_\omega(\tau)$, $\mathrm{FPR}_\omega(\tau)$, and $\mathrm{Precision}_\omega(\tau)$. VUS is then computed by averaging the ROC/PR surfaces over ω and integrating the resulting mean curves:

$$\begin{aligned} \text{VUS-ROC} &= \int \overline{\mathrm{TPR}}(\tau)\, d\,\overline{\mathrm{FPR}}(\tau), \\ \text{VUS-PR} &= \int \overline{\mathrm{Precision}}(\tau)\, d\,\overline{\mathrm{TPR}}(\tau), \end{aligned} \tag{S6}$$

where the overline denotes averaging over $\omega \in \Omega$. In the implementation, τ is swept over 200 uniformly spaced thresholds between the minimum and maximum anomaly scores.

VUS therefore measures how consistently a method ranks anomalies highly under varying temporal tolerance around anomalous ranges. In our implementation, it is a range-aware extension of ROC/PR evaluation rather than an event-counting metric. We report both VUS-ROC and VUS-PR to capture tolerance-aware performance under class imbalance, and we keep the metric implementations fixed across all methods for fair comparison.

F Baselines

F.1 Evaluation Protocol

For the main benchmark in Table 1, methods share data loading, normalization, metric computation, and evaluation procedures. All reported baseline values are obtained from our own reruns, not copied from TAB's public leaderboard. TAB provides model implementations and pipeline components, but our configuration is not an exact reproduction of its leaderboard protocol. Each baseline follows the hyperparameters recommended in its original paper or official implementation, including its window length and method-specific training duration. The 10-epoch setting in Section 4.1 applies only to TENET and is not imposed on the baselines. We use seeds $\{10, 23, 30, 31, 37\}$ for the five-run comparisons, and we do not apply any additional score post-processing (e.g., smoothing or calibration) beyond each method's original score.

For fair comparison, we standardize the evaluation interface across methods. Each baseline produces a scalar anomaly score per time step, and all scores are evaluated by the same metric code. When a model naturally outputs window-level quantities (e.g., reconstructed windows, predicted values, likelihoods, or attention-based scores), we reproduce the anomaly scoring rule of the original method and consistently map it to a point-wise score at the current target time step. Input normalization is performed with `StandardScaler` fitted on the training split and applied to both training and test splits.

Our shared settings differ from TAB's public configuration in four respects. Models requiring validation use a chronological 90/10 train–validation split rather than 80/20. We retain the original dataset releases and keep independent sequences separate rather than concatenating them. Train-fitted standardization is applied to classical as well as deep baselines. For the main benchmark, the VUS tolerance range is fixed to $\Omega = \{0, \ldots, 10\}$ rather than selected separately for each dataset.

Most trainable baselines are evaluated with five independent random seeds and we report seed-averaged performance. For foundation-style models that provide official pre-trained checkpoints, we run the released pre-trained weights and follow the authors' prescribed anomaly detection recipe, rather than re-pretraining from scratch. Overall, model-specific differences are preserved (including window length and scoring rule), while the evaluation code path and metric computation remain identical across methods.

F.2 Code Resources

We evaluate most baselines through the unified **TAB benchmark** codebase[6]. For methods that require dedicated code paths, we use their official implementations.

- **OmniAnomaly** (KDD 2019)[7]
- **SARAD** (NeurIPS 2024)[8]
- **Chronos** (TMLR 2024)[9]
- **CAROTS** (ICML 2025)[10]

[6] https://github.com/decisionintelligence/TAB
[7] https://github.com/NetManAIOps/OmniAnomaly
[8] https://github.com/daidahao/SARAD
[9] https://github.com/amazon-science/chronos-forecasting
[10] https://github.com/kimanki/CAROTS

G System Configuration

All TENET experiments are conducted using the following system configuration.

- **OS:** Ubuntu 24.04.3 LTS.
- **Python:** Python 3.10.19.
- **GPU:** NVIDIA RTX PRO 6000 Blackwell Workstation Edition with 96 GB GPU memory.
- **CPU:** AMD Ryzen 9 9950X3D 16-core processor with 32 hardware threads.
- **System memory:** 91 GiB.
- **CUDA:** CUDA toolkit 12.8.1 and CUDA runtime 12.8.90.
- **PyTorch:** PyTorch 2.11.0+cu128.

H Statistical Significance

Table S3: Standard deviation of anomaly detection performance on four multivariate time series anomaly detection datasets. All metrics are reported in percent and computed over five random seeds. Lower values indicate higher stability across runs.

Dataset	**PSM (25 vars)**				**SMD (38 vars)**				**SWaT (51 vars)**				**SWAN (38 vars)**			
Method	**A-ROC**	**A-PR**	**V-ROC**	**V-PR**	**A-ROC**	**A-PR**	**V-ROC**	**V-PR**	**A-ROC**	**A-PR**	**V-ROC**	**V-PR**	**A-ROC**	**A-PR**	**V-ROC**	**V-PR**
IF	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PCA	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OCSVM	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
DAGMM	2.86	2.05	2.89	2.77	1.96	3.07	2.02	4.83	6.29	2.35	19.03	15.14	4.66	7.57	2.91	5.00
THOC	3.77	5.00	3.73	4.97	3.77	4.58	10.56	9.99	0.95	1.66	15.79	13.57	3.50	5.78	3.39	3.56
DC	0.29	0.24	0.56	0.82	0.11	0.03	0.03	0.25	0.29	0.03	0.10	0.31	0.18	0.08	0.03	0.71
CAROTS	1.21	1.96	2.31	0.07	2.09	5.56	2.52	4.80	1.63	2.21	1.30	1.57	5.80	4.60	2.20	2.50
Omni	1.89	2.34	0.56	0.52	0.11	1.37	7.75	8.23	0.07	0.02	0.11	3.58	0.78	0.72	0.96	0.55
A.T	2.10	0.88	0.95	13.09	1.20	0.18	0.16	13.55	4.39	0.41	0.12	0.42	4.19	1.13	0.07	5.30
PatchTST	0.06	0.10	0.05	0.76	0.33	0.11	0.16	1.09	0.07	0.02	0.02	0.03	0.46	0.38	0.22	0.36
TimesNet	0.56	0.46	0.03	1.40	0.33	0.27	0.16	1.36	0.32	0.09	0.10	0.76	0.36	0.61	0.07	1.20
iTrans	0.11	0.11	0.24	1.02	0.30	0.13	0.04	1.06	0.16	0.01	0.08	0.20	0.29	0.32	0.26	0.58
Modern	0.14	0.18	0.03	0.46	0.80	0.39	0.08	9.74	0.08	0.18	0.08	0.19	0.19	0.33	0.07	0.19
SARAD	1.52	1.26	4.98	1.75	0.42	0.69	0.03	0.12	0.91	12.61	0.56	9.02	1.09	0.90	0.07	0.09
CATCH	0.78	0.75	5.14	0.87	0.18	0.15	0.03	1.60	0.27	0.35	0.16	0.73	0.08	0.10	0.06	0.22
CrossAD	1.30	0.98	0.01	0.66	0.04	0.50	0.01	0.07	3.42	0.91	0.11	0.24	0.12	0.10	0.03	0.04
TTM	0.06	0.13	0.02	0.69	0.41	0.15	0.00	0.00	0.01	0.05	0.07	0.11	0.19	0.16	0.01	0.03
Timer	0.02	0.01	0.02	0.05	0.09	0.32	0.00	0.46	0.01	0.01	0.05	0.06	0.19	0.08	0.01	0.01
Chronos	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
GPT4TS	0.36	0.51	0.07	0.55	0.13	0.15	0.00	0.35	0.42	0.05	0.02	2.75	1.57	1.96	0.08	0.84
TENET	0.69	1.73	0.68	1.70	1.19	1.63	1.19	1.63	0.47	0.96	0.48	0.96	0.96	1.01	0.94	0.72

To assess robustness to random initialization and stochastic optimization, we report the standard deviation of threshold-free ranking metrics across independent runs. Table S3 summarizes the run-to-run variability of AUC-ROC (A-ROC), AUC-PR (A-PR), VUS-ROC (V-ROC), and VUS-PR (V-PR), computed over five random seeds for each trainable method. Lower values indicate higher stability across runs. For TTM, Timer, and GPT4TS, we initialize from released pre-trained weights and fine-tune the models on each dataset before evaluation. Chronos is evaluated using released pre-trained weights, but we do not report standard deviations because repeated runs were not conducted for this model. Dashes indicate cases where repeated-run standard deviations are unavailable.

I One-Class Objective Configurations

This appendix specifies the implementation details of the one-class objectives used in the encoder–objective comparisons in Table 2 and Table S4. For a fixed encoder, all four objectives receive the same pooled current-point state $\mathbf{c} \in \mathbb{R}^h$ with $h = 256$. Objective-specific scoring modules and losses differ, while the optimizer (AdamW), batch size, learning rate, and number of epochs follow the corresponding NLL experiment for the same encoder and dataset. This keeps the encoder interface and main training protocol comparable across one-class objectives.

NLL (standard normal reference). The NLL head is parameter-free. For each pooled state $\mathbf{c}$, the per-sample loss and anomaly score are

$$\mathcal{L}_{\mathrm{NLL}}(\mathbf{c}) = \tfrac{1}{2}\big(\|\mathbf{c}\|_2^2 + h\log 2\pi\big), \qquad s_{\mathrm{NLL}}(\mathbf{c}) = \tfrac{1}{2}\big(\|\mathbf{c}\|_2^2 + h\log 2\pi\big).$$

Training minimizes the mean of $\mathcal{L}_{\mathrm{NLL}}$ over normal training windows, which corresponds to fitting the pooled state to a fixed standard normal reference $\mathcal{N}(\mathbf{0}, \mathbf{I}_h)$ without learning any additional parameters.

DeepSVDD (fixed-center one-class). A center vector $\boldsymbol{\mu} \in \mathbb{R}^h$ is computed once, before training, by averaging encoder outputs over the first 50 training batches with the encoder in evaluation mode. Following the original protocol of Ruff et al. [34], components of $\boldsymbol{\mu}$ with $|\mu_i| < 0.01$ are clamped to 0.01 to avoid the trivial solution at the origin. $\boldsymbol{\mu}$ is then registered as a fixed buffer. For each pooled state, the loss and score are

$$\mathcal{L}_{\mathrm{SVDD}}(\mathbf{c}) = \|\mathbf{c} - \boldsymbol{\mu}\|_2^2, \qquad s_{\mathrm{SVDD}}(\mathbf{c}) = \|\mathbf{c} - \boldsymbol{\mu}\|_2^2.$$

Only the encoder parameters are updated during training, and $\boldsymbol{\mu}$ is not updated.

THOC. THOC uses multiple hyperspheres and hypersphere centers to define a one-class objective [40]. In our implementation, a set of $K = 8$ learnable hypersphere centers $\mathbf{P} \in \mathbb{R}^{K\times h}$ is maintained on the unit sphere by re-normalizing each center to unit ℓ_2 norm at every forward pass. For pooled state $\mathbf{c}$, we compute soft assignments to the centers and a weighted cosine distance:

$$\mathbf{R} = \mathrm{softmax}\big(\hat{\mathbf{c}}\,\mathbf{P}^\top\big) \in \mathbb{R}^K, \qquad s_{\mathrm{THOC}}(\mathbf{c}) = \sum_{k=1}^{K} R_k\big(1 - \cos(\mathbf{c}, \mathbf{P}_k)\big),$$

where $\hat{\mathbf{c}} = \mathbf{c}/(\|\mathbf{c}\|_2 + \epsilon)$, with a small ϵ for numerical stability. The training loss combines the mean of s_{THOC} with an orthogonality regularizer on the hypersphere centers:

$$\mathcal{L}_{\mathrm{THOC}} = \frac{1}{B}\sum_{b=1}^{B} s_{\mathrm{THOC}}(\mathbf{c}_b) + \lambda_{\mathrm{orth}}\left\|\mathbf{P}\mathbf{P}^\top - \mathbf{I}_K\right\|_F^2.$$

We set $\lambda_{\mathrm{orth}} = 1.0$ in all objective-comparison experiments. Both encoder and hypersphere-center parameters are jointly optimized.

DAGMM (GMM-density one-class). A compression network $g_\psi\colon \mathbb{R}^h \to \mathbb{R}^{16}$ produces a low-dimensional latent $\mathbf{z}_c = g_\psi(\mathbf{c})$. We then form an extended representation $\mathbf{z} \in \mathbb{R}^{18}$ by appending $\|\mathbf{c}\|_2$ and $\|\mathbf{z}_c\|_2$ to $\mathbf{z}_c$, mirroring the reconstruction-based norm features used by Zong et al. [56]. An estimation network with a softmax output predicts soft GMM assignments $\boldsymbol{\gamma} \in \mathbb{R}^{n_{\mathrm{gmm}}}$ for $n_{\mathrm{gmm}} = 4$ components. GMM parameters $(\boldsymbol{\pi}, \boldsymbol{\mu}_{1:n_{\mathrm{gmm}}}, \boldsymbol{\Sigma}_{1:n_{\mathrm{gmm}}})$ are stored as buffers and updated at training time using an exponential moving average with factor 0.01, computed from the detached $(\mathbf{z}, \boldsymbol{\gamma})$ to avoid backpropagating through the GMM update. The per-sample training loss and inference score are the negative log-likelihood under the diagonal-covariance GMM,

$$\mathcal{L}_{\mathrm{DAGMM}}(\mathbf{c}) = s_{\mathrm{DAGMM}}(\mathbf{c}) = -\log\sum_{k=1}^{n_{\mathrm{gmm}}} \pi_k\,\mathcal{N}\big(\mathbf{z} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\big),$$

where each $\boldsymbol{\Sigma}_k$ is regularized with a diagonal $10^{-3}\mathbf{I}$ for numerical stability. Encoder, compression, and estimation parameters are jointly optimized, while GMM buffers are updated via the EMA rule.

J Additional Experiments

J.1 Additional Encoder–Objective Comparison

Table S4: Encoder and one-class objective comparison on SWAN. All metrics are reported in percent and averaged over five random seeds. **Bold** indicates the best score.

Objective	NLL				DeepSVDD				THOC				DAGMM			
Metric	A-ROC	A-PR	V-ROC	V-PR	A-ROC	A-PR	V-ROC	V-PR	A-ROC	A-PR	V-ROC	V-PR	A-ROC	A-PR	V-ROC	V-PR
LSTM	73.71	64.50	64.30	72.30	78.99	70.41	68.64	76.06	58.26	47.26	55.39	63.39	75.11	68.32	59.33	79.04
Dilated RNN	68.00	57.53	62.11	69.24	79.59	72.75	67.87	76.25	64.48	55.54	60.81	67.86	78.21	71.73	61.55	78.05
Mamba	52.75	36.66	51.85	57.72	82.70	66.93	70.17	73.81	75.18	57.75	64.85	68.22	68.81	51.94	62.21	66.58
ModernTCN	63.38	54.01	58.21	73.65	63.37	53.97	58.14	73.71	63.40	47.90	61.43	66.45	62.01	52.19	57.56	72.38
GPT4TS	80.46	71.79	66.61	77.42	75.89	65.20	61.87	73.64	78.68	69.41	57.97	78.78	47.93	39.02	48.36	58.76
TENET	**88.36**	**84.07**	**72.98**	**80.97**	**89.01**	**85.18**	**73.60**	**81.63**	**88.67**	**84.46**	**73.51**	**81.35**	**84.39**	**79.21**	**71.10**	**79.31**

We further examine the encoder–objective trend by repeating the same comparison on SWAN. The protocol, metrics, objectives, and pooled-state dimensionality are kept unchanged across encoders. This setting examines whether TENET remains effective when the same representation is scored by different one-class objectives.

Table S4 shows that TENET obtains the best score across all reported objective–metric pairs on SWAN. On A-PR, TENET exceeds the next-best encoder by more than 13 percentage points under both NLL and DeepSVDD. Together with the PSM results in Table 2, this provides additional evidence that TENET's advantage is not specific to the default NLL score. Rather, the results support our claim that constructing an informative current-point state remains important across different one-class objectives.

J.2 Fixed Versus Learnable Gaussian References

We also evaluate a learnable diagonal Gaussian reference on PSM and SMD, with score

$$\mathcal{A}_{\mathrm{learn}}(\mathbf{c}) = \frac{1}{2}\sum_{j=1}^{h}\left[\frac{(c_j-\mu_j)^2}{v_j} + \log v_j + \log(2\pi)\right].$$

The means and variances are jointly optimized with TENET, initialized to zero and one, respectively. We parameterize $v_j = \mathrm{softplus}(a_j) + 10^{-6}$ to ensure positive variances. The training settings and five seeds match the fixed-reference experiments. The fixed-reference results in Table S5 are taken from Table 1.

Table S5: Gaussian reference comparison for **TENET** on PSM and SMD. Metrics are reported in percent as mean ± standard deviation over five seeds.

Reference	PSM				SMD			
	A-ROC	A-PR	V-ROC	V-PR	A-ROC	A-PR	V-ROC	V-PR
Fixed standard normal	86.14 ± 0.69	74.27 ± 1.73	85.82 ± 0.68	74.21 ± 1.70	90.04 ± 1.19	74.78 ± 1.63	89.50 ± 1.19	74.41 ± 1.63
Learnable Gaussian	86.20 ± 0.74	74.44 ± 1.68	85.88 ± 0.73	74.38 ± 1.65	90.03 ± 0.97	74.77 ± 1.27	89.50 ± 0.94	74.42 ± 1.20

The reported differences are small relative to variation across runs and show no consistent advantage for the learnable reference. We therefore retain the fixed standard normal reference for simplicity, while TENET also supports learning the reference parameters.

J.3 Cross-Model Computational Cost

Table S6 compares representative baselines with TENET on synthetic inputs with $N = L = 100$, using FP32 precision on the same GPU. Inference latency is measured at batch sizes 1 and 64 and reported per window. Training uses batch size 64 and each model's training objective. For the baselines, training-step time is estimated from the difference between four-epoch and one-epoch runtimes divided by the additional optimization steps. TENET is timed over warmed-up optimization steps. Parameter storage and peak allocated training memory are reported in decimal MB. Operation counts use `fvcore`, counting one multiply–accumulate as one operation, with analytical recurrent-layer counts added for operators not traced by the profiler.

Table S6: Cross-model computational cost at $N = L = 100$. Inference latency is per window, and training latency is per step. B denotes batch size.

Method	**Parameters (MB)**	**Operations (G)**	**Inference $B = 1$ (ms)**	**Inference $B = 64$ (ms)**	**Training $B = 64$ (ms)**	**Training Peak $B = 64$ (MB)**
THOC	0.53	0.033	1.67	0.0267	3.63	439.07
OmniAnomaly	1.42	0.035	0.31	0.0059	2.05	224.62
Anomaly Transformer	19.82	0.526	0.85	0.0420	16.51	3,092.85
CATCH	4.39	0.665	1.69	0.4029	85.31	9,441.99
SARAD	38.45	0.986	0.44	0.0581	12.56	1,134.95
DCdetector	3.71	1.848	1.35	0.2530	53.93	8,183.42
CrossAD	10.58	4.133	0.78	0.5267	132.62	14,654.57
TENET	700.49	3.151	33.18	1.2604	294.09	8,761.62

TENET has the largest parameter storage and higher measured latency in this comparison, reflecting the cost of its separate temporal encoders. Its peak training memory is lower than that of CATCH and CrossAD, but higher than that of the other listed baselines. Together with the detection results, this comparison makes the performance–cost trade-off explicit rather than suggesting that TENET is uniformly more efficient.

K Representation Geometry Diagnostics

This appendix defines the diagnostics used in Figure 2. All diagnostics are computed from the pooled system state $\mathbf{c} = f_\Theta(\mathbf{X}) \in \mathbb{R}^h$. The covariance-based diagnostics use normal training windows to inspect the learned state geometry, while the radial and directional probes evaluate how much anomaly-discriminative information remains in different components of the state.

Covariance Spectrum. Let $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_B]^\top$ be a batch of pooled states and let $\bar{\mathbf{c}} = B^{-1} \sum_{i=1}^{B} \mathbf{c}_i$. We compute the empirical covariance

$$\Sigma = \frac{1}{B-1} \sum_{i=1}^{B} (\mathbf{c}_i - \bar{\mathbf{c}})(\mathbf{c}_i - \bar{\mathbf{c}})^\top.$$

Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_h \geq 0$ be the eigenvalues of Σ. This spectrum describes how the representation variance is distributed across latent directions. A spectrum concentrated in a few large eigenvalues indicates that the state vectors occupy a low-dimensional subspace, whereas a flatter spectrum indicates a more distributed representation.

Effective Rank. Effective rank summarizes the dimensionality of the covariance spectrum [33]. With normalized eigenvalues $p_k = \lambda_k / \sum_{j=1}^{h} \lambda_j$, it is defined as

$$\mathrm{eRank}(\Sigma) = \exp\left(-\sum_{k=1}^{h} p_k \log p_k \right).$$

Higher effective rank means that variance is spread across more latent directions. In Figure 2(a), a higher value therefore indicates a less low-rank, less degenerate state geometry during training.

Cumulative Explained Variance Ratio. The cumulative explained variance ratio (EVR) shows how quickly the covariance spectrum concentrates. For the top m components, we compute

$$\mathrm{EVR}(m) = \frac{\sum_{k=1}^{m} \lambda_k}{\sum_{j=1}^{h} \lambda_j}, \qquad m = 1, \ldots, h.$$

If $\mathrm{EVR}(m)$ rises quickly, then a small number of principal directions accounts for most of the state variance. If it rises slowly, the representation uses a broader portion of the state space. Figure 2(b) reports this curve after training.

Directional Probe. To test whether the state direction contains anomaly-relevant information, we construct a direction-only score. For each state, define the unit direction

$$\mathbf{u}_i = \frac{\mathbf{c}_i}{\|\mathbf{c}_i\|_2 + \epsilon}.$$

Here ϵ is a small constant used only for numerical stability. We compute the mean normal direction from training states,

$$\bar{\mathbf{u}} = \frac{\frac{1}{B} \sum_{i=1}^{B} \mathbf{u}_i}{\left\| \frac{1}{B} \sum_{i=1}^{B} \mathbf{u}_i \right\|_2 + \epsilon},$$

and score a test state by cosine distance to this direction:

$$s_{\mathrm{dir}}(\mathbf{c}) = 1 - \frac{\mathbf{c}^\top \bar{\mathbf{u}}}{\|\mathbf{c}\|_2 + \epsilon}.$$

This score removes radial magnitude and keeps only angular deviation from the normal direction. Figure 2(c) reports V-PR using this directional score.

Radial Probe. The norm-only score isolates radial information:

$$s_{\mathrm{norm}}(\mathbf{c}) = \|\mathbf{c}\|_2.$$

This score ignores angular information and uses only the distance of the state from the origin under the fixed standard normal reference. Figure 2(d) reports V-PR using this radial score.

L Training and Test-Time Perturbation Sensitivity

L.1 Training Contamination

We evaluate whether TENET remains stable when perturbed windows are introduced into the normal training set. At each optimization step, a random fraction of training windows, with ratio in $\{0, 1, 3, 5, 10, 20\}\%$, is replaced by a perturbed version of the same window. In this experiment, evaluation uses the original test split without added perturbations, and all other hyperparameters follow the main experimental protocol.

Zero-Mean Perturbations. We first consider per-point zero-mean perturbations using Gaussian noise $\mathcal{N}(0, \sigma^2)$ and variance-matched Uniform noise $\mathcal{U}(-\sigma\sqrt{3}, \sigma\sqrt{3})$ with $\sigma = 3.0$. Figures S2 and S3 show that TENET remains relatively stable under these perturbations. Performance changes modestly as the perturbation ratio increases, with drops remaining within roughly 1.6%pt of the clean baseline even at the 20% perturbation rate. The Gaussian and uniform settings also produce similar trends. These results suggest that the learned time-point representation is less sensitive to point-wise noise injected during training.

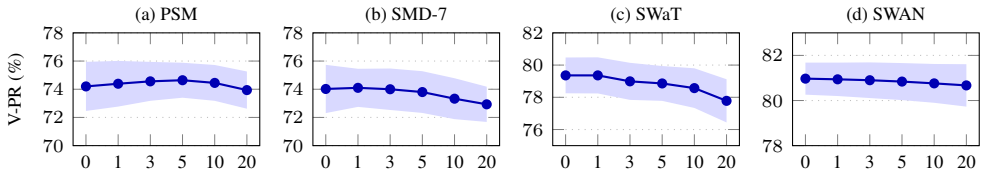

Figure S2: Gaussian training perturbation robustness. Panels report V-PR (%) versus perturbation rate (%) across datasets, with mean and ± 1 standard deviation over five seeds.

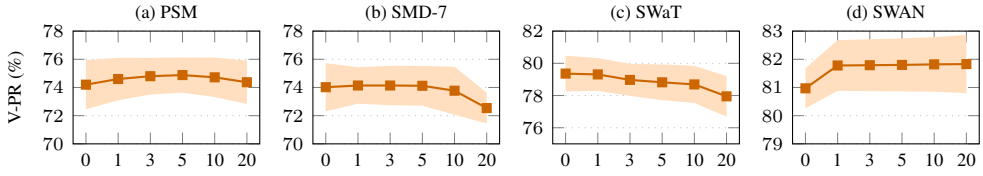

Figure S3: Uniform training perturbation robustness. Panels report V-PR (%) versus perturbation rate (%) across datasets, with mean and ± 1 standard deviation over five seeds.

Coherent Offset Perturbations. We also test a more structured perturbation in which each selected window receives a per-variable constant offset drawn from $\mathcal{N}(0, \sigma^2)$. Unlike point-wise zero-mean noise, this perturbation shifts an entire variable trajectory within a window and is therefore closer to calibration drift or a constant sensor offset. Figure S4 shows that offset perturbations are more challenging, especially on SMD and SWaT, where V-PR drops by up to roughly 4.9%pt at the 20% perturbation rate. SWAN remains comparatively stable under the same perturbation. Even in this more challenging setting, TENET retains useful detection performance, suggesting that the learned current-point representation remains stable under moderate training-set shifts.

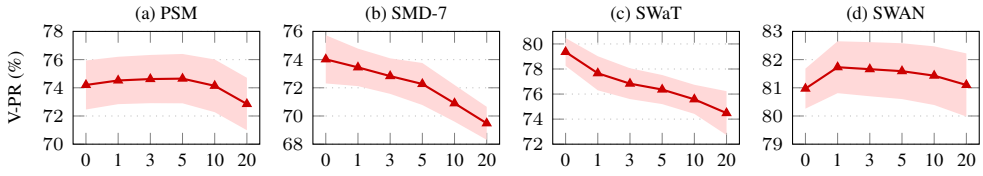

Figure S4: Constant-offset training perturbation robustness. Panels report V-PR (%) versus perturbation rate (%) across datasets, with mean and ± 1 standard deviation over five seeds.

L.2 Current-Point Perturbations at Inference

Because the current observation determines the temporal query, perturbing it can change the retrieved context. To measure this sensitivity, we add independent zero-mean Gaussian noise with standard deviation σ to each variable of the current observation $\mathbf{x}_t$ in every test window on PSM. Noise is added after standardization, while the preceding observations and original evaluation labels remain unchanged. The model is not retrained, so this experiment treats the added perturbation as nuisance noise rather than relabeling it as an anomaly. Figure S5 reports results over five seeds.

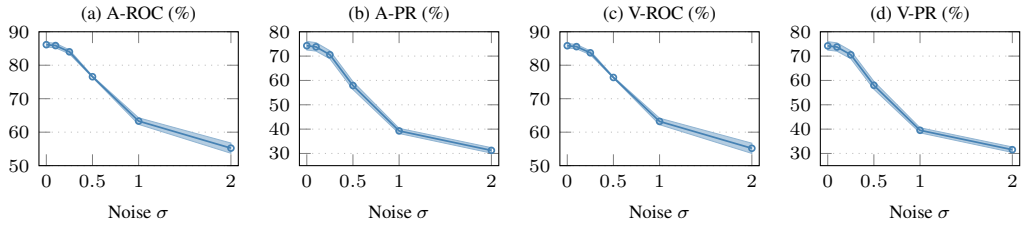

Figure S5: Sensitivity of TENET to test-time current-point noise on PSM. Panels report each metric (%) versus noise standard deviation σ. Lines and markers show the mean over five seeds, and shaded bands indicate ± 1 standard deviation.

Performance changes little at $\sigma = 0.1$, but decreases as the noise grows, with substantial degradation at higher noise levels. These results show tolerance to small current-point perturbations, not invariance to arbitrary corruption.

M Broader Impacts

This work studies unsupervised anomaly detection for multivariate time series, with potential applications in industrial monitoring, server operations, scientific instruments, and cyber-physical systems. Improved detection can help operators identify abnormal system behavior earlier, reduce downtime, and support more reliable maintenance decisions. By showing that one-class TSAD can be a strong alternative when paired with appropriate representation construction, this work broadens the methodological choices available for both practical applications and future TSAD research. The representation-focused design of TENET may also be useful as a general state encoder for downstream monitoring tasks beyond binary anomaly scoring.

At the same time, anomaly detectors should be deployed with care. False positives can increase operator burden, while false negatives can delay intervention during real faults. TENET provides anomaly scores rather than automated corrective actions, so practical deployments should combine the model with domain-specific thresholds, human review, and post-deployment monitoring. Because real systems often experience distribution shift, calibration should be revisited after changes in sensors, operating policies, or system configuration.

The method is evaluated on public benchmark datasets and synthetic robustness settings. These benchmarks are useful for controlled comparison, but they do not cover every deployment scenario.

N Limitations and Future Work

TENET is designed to isolate the effect of current-point system state construction under a simple one-class scoring rule. This design makes the empirical analysis cleaner, but it also leaves several limitations.

First, TENET uses inter-variable self-attention to compose variable-level states. This allows flexible cross-variable modeling, but its cost grows with the number of variables. The scalability study shows that the current implementation remains practical for the tested regimes, yet larger industrial systems may require sparse, hierarchical, or graph-constrained inter-variable layers. Future work should therefore develop scalable inter-variable composition modules that preserve current-point-centric state construction while reducing the cost of multivariate composition in large sensor systems. For long-window training, smaller micro-batches with gradient accumulation and activation checkpointing can reduce memory requirements. Compressing long histories through downsampling or shorter key–value sequences is another possible direction for future work, but its effect on detection performance remains to be evaluated.

Second, a deep theoretical explanation of why TENET works well remains open. The experiments with multiple one-class objectives show that TENET's advantage is not tied to a particular scoring rule and provide strong evidence for the utility of its architecture. A deep theoretical explanation of why TENET produces an effective anomaly-ranking structure for one-class MTSAD would be a valuable direction for future work.

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 claims and scope are stated in the abstract and Introduction, supported by the experimental evidence in Section 4.2, and bounded by the discussions in Appendix M and Appendix N.

   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: Appendix N discusses the remaining limitations and future work.

   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 does not present formal theoretical theorems or proof-based claims. Section 3 provides the formulation and equations rather than formal theoretical results.

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 manuscript provides the information needed to reproduce the reported experiments in the method and experimental setup, with implementation pseudocode and supporting details in Appendices C, D, E, F, and G.

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: [No]

   Justification: Appendix D provides dataset access information and Appendix F lists external code resources. The source code for the proposed method cannot be distributed with the current submission because any external code release, including supplementary or anonymous distribution, requires internal open-source approval. We plan to release an official repository after completing this approval process.

   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: Section 4.1 specifies the main training and evaluation settings. Appendix D gives data split information, and Appendix F describes preprocessing, score mapping, and evaluation protocol.

   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: [Yes]

   Justification: Main tables report averages over repeated runs, and Appendix H reports standard deviations for trainable methods.

   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: [Yes]

   Justification: Appendix G reports the hardware and software configuration used for the experiments. Section 4.6 further reports latency and memory scaling.

   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 uses documented datasets, does not involve human subjects, and follows the NeurIPS Code of Ethics.

   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: Appendix M discusses potential positive impacts, possible negative impacts, and practical mitigation considerations.

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, scraped dataset, or pre-trained model with dual-use risks of the type described in the 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: Appendix D credits data sources and reports access terms where available, and Appendix F credits external code and model resources used in the experiments.

    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: [N/A]

    Justification: The current manuscript does not release a new dataset, benchmark, model checkpoint, or public software asset.

    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 research does not involve crowdsourcing experiments or 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: The research does not involve crowdsourcing experiments or human subjects, so IRB approval is not applicable.

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: LLMs are not an important, original, or non-standard component of the method, experiments, or scientific claims.

    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.