Deep2DO: Deep Decoupled Diagnostic Observer for Unsupervised Anomaly Detection

Assmaa Alsamadi (✉), Fannia Pacheco, and Paul Honeine

Univ Rouen Normandie, LITIS UR 4108, F-76000 Rouen, France
assmaa.alsamadi@univ-rouen.fr

Abstract. Anomaly detection (AD) in dynamic systems has attracted significant research interest, leading to the development of high-performance mathematical models. Among these, diagnostic observers (DOs) rely on modeling explicit representations of the system. In contrast, Deep Learning (DL) models excel at learning representations from data for AD. Yet, they often require costly training or fine-tuning of large architectures to generalize across datasets. This paper introduces Deep2DO, the first unsupervised AD model that integrates the dynamics of DO into a DL framework. This decouples DO from explicit system modeling through a DL re-parameterization. Furthermore, DOs impose input and output availability constraints that are frequently unmet in time series AD. Our approach addresses this by correctly estimating unknown DO inputs through a systematic definition process, yielding a tractable and optimally structured deep DO formulation. Comprehensive experiments on large time series benchmarks demonstrate that our proposal outperforms more than 40 state-of-the-art models, exhibiting strong generalization across diverse anomaly types in both univariate and multivariate time series.

Keywords: Diagnostic Observer · Deep Learning · Anomaly Detection · Time series.

1 Introduction

The widespread use of digital devices has led to vast amounts of time series data across domains [11]. Time series data often include rare anomalies that deviate from normal patterns. Anomaly detection (AD) aims to identify such deviations and is widely used in fraud detection and predictive maintenance [3,38]. Anomalies may occur at a single point (called point anomaly) or across multiple consecutive observations (called sequence anomaly) [24].

Machine Learning (ML) is suitable for AD due to its ability to learn complex patterns and capture sequential dependencies. ML-based AD models can learn in different settings, mainly supervised and unsupervised learning [34]. Prior to the rise of ML, numerous mathematical frameworks for AD in dynamic systems were developed using observer-based models and their diagnostic observer (DO) variants [30,12]. Most of these models detect anomalies using a residual error as

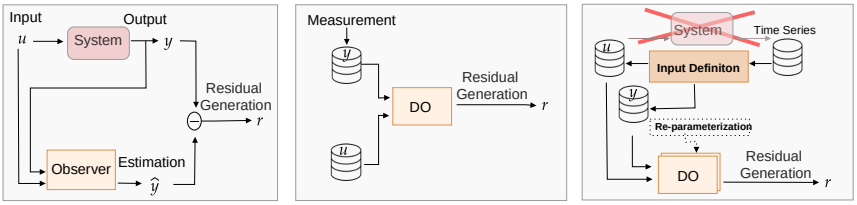

Fig. 1: Three observer-based AD strategies where an SSM estimates the output and generates a residual r anomaly score. **a) Observer-based:** The input u and output y are available, and the SSM is constructed from a mathematical model of the system. **b) Data-driven:** Measured input–output data (u, y) are used to design DO without explicit dynamic modeling. **c) Proposed Deep2DO:** Only time-series data are available. A DL block extracts (u, y), and SSM_{DO} is reparameterized and optimized for AD.

an anomaly score, where the system's output is estimated and then compared with the actual output. As illustrated in Fig. 1(a) [7], a classic observer models the normal dynamics of the system and generates a fault-free estimate $\hat{y}$ that closely approximates the actual value y that is accessible from a known defined dynamic system. As an alternative, Fig. 1(b) illustrates a data-driven approach [6]. This approach utilizes real measured data to adjust and define the DO's parameters that generate a fault-free estimate. These approaches require access to the modeled system and its input-output pair. However, modeling the system can be challenging and inaccurate, as it often overlooks unmodeled external factors.

To decouple DO design from explicit system modeling, we propose Deep2DO, a deep learning (DL) framework for unsupervised time series AD. As illustrated in Fig. 1(c), Deep2DO is composed of two main processes that decouple DO from system modeling: **1)** an input definition that estimates the inaccessible system input u and embeds the series into a higher-dimensional space D to form (u, y); 2) A re-parameterization to obtain DO parameters in the absence of system modeling. DO is instantiated in D feature depth whose residuals are aggregated into a global anomaly score. We further introduce a stacked version with N sequential layers to progressively refine anomaly representations. The main contributions of this paper are:

- A Deep2DO model, the first approach to integrate the DO's mathematical structure within a DL framework that decouples the DO from modeling through a re-parameterization.
- Two straightforward input estimation methods for DO's unknown inputs.
- A stacked Deep2DO architecture that effectively models anomaly information.
- Extensive experiments showing that Deep2DO outperforms more than 40 state-of-the-art models on large real-world and benchmark time series datasets.

2 Related Work

Unsupervised Time Series Anomaly Detection. A discrete time series $y(k)$ consists of T ordered observations $y(1), \ldots, y(T)$, with $y(k) \in \mathbb{R}^d$. It is univariate if $d = 1$ and multivariate if $d > 1$. AD is often performed in unsupervised settings to assign a label $l(y(k)) \in \{0, 1\}$, where 0 denotes normal and 1 an anomaly [21]. Traditional methods are statistical-based approaches that highlight anomalies based on deviations in low-dimensional data [35,5], while recent DL approaches detect anomalies via reconstruction errors, either by predicting future values or reconstructing anomaly-free inputs using Autoencoders, TimesNet, or CrossAD [41,33,43,20]. More recently, the previous tasks are handled by foundation models that leverage pretraining on unlabeled data to capture long-term temporal dependencies such as OFA and Moment, and among others [48,14].

Dynamic System Modeling. A continuous dynamic system with input $u(t) \in \mathbb{R}^D$ and output $y(t) \in \mathbb{R}^d$ can be modeled via state-space models (SSMs). An SSM captures temporal evolution through a linear relationship between $u(t)$ and $y(t)$ through a state vector $x(t) \in \mathbb{R}^n$ using the system matrices $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ as in (1) [30]. For digital simulation, the modeled SSM is discretized using zero-order hold or bilinear transforms to obtain $(\overline{\mathbf{A}}, \overline{\mathbf{B}}, \overline{\mathbf{C}}, \overline{\mathbf{D}})$ as in (2).

$$\begin{cases} \dot{x}(t) = \mathbf{A}x(t) + \mathbf{B}u(t), \\ y(t) = \mathbf{C}x(t) + \mathbf{D}u(t), \end{cases} \quad (1) \qquad \mathrm{SSM}_{\overline{\mathbf{A}},\overline{\mathbf{B}},\overline{\mathbf{C}},\overline{\mathbf{D}}} : \begin{cases} x(k) = \overline{\mathbf{A}}x(k-1) + \overline{\mathbf{B}}u(k) \\ y(k) = \overline{\mathbf{C}}x(k) + \overline{\mathbf{D}}u(k) \end{cases} \quad (2)$$

Analogous DL architectures to (2) capture long-range dependencies from historical data[16,15]. The SSM matrices are learned as in S4 [16] or Mamba [15]. Their recurrent form (2) can be reformulated convolutionally as $y = \mathcal{K} \odot u$ to avoid sequential computation of x [16]. SSMs are also combined with convolutional and attention layers for time series forecasting and image AD [42,17].

In addition, for a given $\mathrm{SSM}_{\overline{\mathbf{A}},\overline{\mathbf{B}},\overline{\mathbf{C}},\overline{\mathbf{D}}}$ and $y \in \mathbb{R}^{d \times T}$, u can be recovered from y over T steps if the SSM satisfies [32, Theorem 2, Corollary 1]:

Theorem 1 (Sain-Massey Condition [32]) *An SSM is invertible over T steps iff* $\mathrm{rank}(\mathbf{M}_T) - \mathrm{rank}(\mathbf{M}_{T-1}) = D$,

$$\textit{with } \mathbf{M}_i = \begin{cases} \overline{\mathbf{D}} & \textit{if } k = j, \\ \overline{\mathbf{C}}\overline{\mathbf{A}}^{(k-j-1)}\overline{\mathbf{B}} & \textit{if } k > j, \\ 0 & \textit{if } k < j, \end{cases} \qquad k, j = 1, \ldots, i.$$

However, this operation has no DL equivalent. Only data-driven procedures exist with system matrices subject to hard-to-enforce constraints [31].

Diagnostic Observer. A discrete Diagnostic Observer (DO) is commonly used for fault modeling in dynamic systems. It maps pair input-output $(u(k), y(k))$ to a latent observer state $z(k) \in \mathbb{R}^s$ of order s, to estimate a fault-free output

and generate a residual $r(k) \in \mathbb{R}$. High variation in $r(k)$ indicates the presence of an anomaly [12,7]. The discrete DO is defined as follows:

$$\text{DO} = \begin{cases} z(k+1) = \mathbf{G}z(k) + \mathbf{H}u(k) + \mathbf{L}y(k) \\ r(k) = vy(k) - wz(k) - qu(k) \end{cases} \tag{3}$$

where $v \in \mathbb{R}^{1\times d}$, $w \in \mathbb{R}^{1\times s}$, $q \in \mathbb{R}^{1\times D}$, $\mathbf{G} \in \mathbb{R}^{s\times s}$, $\mathbf{H} \in \mathbb{R}^{s\times D}$, $\mathbf{L} \in \mathbb{R}^{s\times d}$ and vy is estimated by $wz - qu$.

Observers are widely used in dynamic systems for AD [30,8,7,6]. A classical observer depends on modeling the system via an SSM. This SSM provides access to the system's input–output, allowing the observer to estimate fault-free outputs and compare them with the real outputs through residuals, which serve as anomaly scores [30,8]. In contrast, data-driven DOs rely on measured data to model system dynamics [7,6]. This system provides input-output pairs used by the DO to generate residuals for anomaly scoring. Both approaches rely heavily on system modeling and input-output data.

3 Methodology

3.1 Problem Formulation

A DO for AD in time series when only y is available requires recovery of the DO input u and modeling the system dynamics to define DO's parameters.

DO Input. u is typically reconstructed via system modeling. For low-dimensional outputs $y = \text{SSM}_{\overline{\mathbf{A}},\overline{\mathbf{B}},\overline{\mathbf{C}},\overline{\mathbf{D}}}(u)$ [28], and the input u is recovered since the SSM is invertible. For high-dimensional outputs, consider an SSM of y with $D = d$ and defined by the matrices:

$$\overline{A}_{i,j} = \begin{cases} I_D, & j = i+1, j \in \{1, \cdots, l\} \\ 0, & \text{otherwise.} \end{cases}, \overline{\mathbf{B}} = \begin{bmatrix} I_D \\ \cdots \\ 0 \end{bmatrix}, \overline{\mathbf{C}} = \begin{bmatrix} I_D \cdots 0 \end{bmatrix} \text{ and } \overline{\mathbf{D}} = I_D,$$

where $\overline{\mathbf{A}} \in \mathbb{R}^{n\times n}$ (with $n = lD > d$) consists of l blocks $\overline{A}_{i,j}$, each being either the identity matrix $I_D \in \mathbb{R}^{D\times D}$ or the zero matrix $0 \in \mathbb{R}^{D\times D}$, and with input u with dimension d (i.e., $D = d$). This SSM satisfies Theorem 1 since $\overline{\mathbf{C}}\overline{\mathbf{A}}^k\overline{\mathbf{B}} = I_D$ for $k = 0$ and 0 otherwise, ensuring that the SSM is invertible and u can be recovered from y. Moreover, the inverse of an SSM is itself an SSM [31], allowing u to be expressed as

$$u = \text{SSM}_{\overline{\mathbf{A}'},\overline{\mathbf{B}'},\overline{\mathbf{C}'},\overline{\mathbf{D}'}}(y). \tag{4}$$

However, finding the matrix parameters that best fit the data, as in [31], becomes infeasible in high-dimensional spaces due to numerical instability and the lack of theoretical guarantees for the matrices in (4) and [28], whose uniqueness and optimality cannot be ensured. To address this, we propose approximating the SSM in (4) using a DL-based input definition process, which reconstructs the input u from a time series z_0.

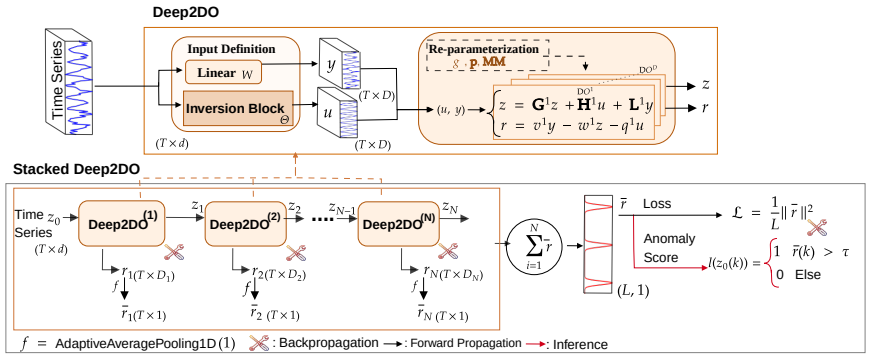

Fig. 2: Schematic illustration of the proposed Deep2DO (top) and its stacked implementation Stacked Deep2DO (bottom). The latter sequentially implements N Deep2DO layers, where each layer generates a residual r_i that is used for both loss computation during training and anomaly scoring during inference.

DO Parameters. The DO's matrices in (3) can be calculated through an algebraic design [46,7] that assumes that the dynamic system is given by an SSM, i.e., $y(k) = \text{SSM}_{\overline{\mathbf{A}},\overline{\mathbf{B}},\overline{\mathbf{C}},\overline{\mathbf{D}}}(u(k))$ with $\overline{\mathbf{A}}, \overline{\mathbf{B}}, \overline{\mathbf{C}}, \overline{\mathbf{D}}$ are known and relies on a parity vector $\mathbf{p} \in \mathbb{R}^s$ to define the matrices ($\mathbf{G}$, $\mathbf{H}$, $\mathbf{L}$) and vectors (v, w, q). For instance, $\mathbf{H}$ is defined as $\mathbf{H}_{i,j} = f_{\mathbf{H}}(\mathbf{p}, i, j)$ where $f_{\mathbf{H}}$ is a function that describes how the values of $\mathbf{p}$ are distributed across rows and columns. A similar procedure is applied to obtain $\mathbf{L}, v$ and q. The parity vector $\mathbf{p}$ is computed as a null space as:

$$\mathbf{p}\left[\overline{\mathbf{C}}\ \overline{\mathbf{C}\mathbf{A}}\ \ldots\ \overline{\mathbf{C}\mathbf{A}}^{\mathbf{s}}\right]^T = \mathbf{0}, \text{ with } \mathbf{p} = \left[p_0\ p_1\ \cdots\ p_s\right] \text{ and } p_i \in \mathbb{R}. \tag{5}$$

The DO provides a strong mathematical basis for AD but suffers from parity vector selection and high computational cost. To address this, we propose a DL-based DO that learns the parity vector through optimization. Section 3.2 explains the model design that addresses DO's AD formulation for time series, where the DO structure serves as an inductive bias for AD, rather than as a strict implementation of classical DO theory.

3.2 Model Design

We propose Deep2DO, the first DL–based DO framework for time series AD that decouples DO from system modeling and relies on two major contributions: i) Input Definition, where the time series provides the system output y, from which the input u is recovered via an inversion. ii) Re-parameterization, which constructs the DO matrices without an explicit system model. A single Deep2DO outputs an embedding z that preserves anomaly-related information, and a residual r used for anomaly scoring. We further introduce a stacked Deep2DO implementation as illustrated in Fig. 2. We stack N Deep2DO layers, where z_i is fed to the $(i+1)$-th block. Residuals are adaptively averaged within

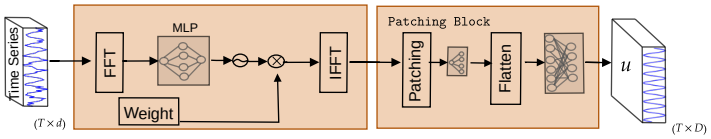

Fig. 3: FMLP architecture to recover the input u

each block and summed to obtain the final residual. During training, the stacked Deep2DO learns normal patterns keeping the residual close to zero for normal data. During inference, the residual exhibits spikes at anomalous points. The loss is the temporal average of the final residual. Deep2DO and its stacked version are described in Section 3.3 and Section 3.4 respectively.

3.3 Deep2DO

The Deep2DO layer implements (3) in a DL framework for time series AD. Its input (u, y) is defined via the Input Definition, and matrices $(\mathbf{G}, \mathbf{H}, \mathbf{L}, q, v, g)$ are obtained via Re-parameterization as explained below.

Input Definition. A Deep2DO recovers an input u from a time series z_0 via a DL-based inversion architecture, either the `CNet` or the `FMLP` as follows:

- `CNet` Architecture. A 1-D convolutional network inspired by the fact that there exists a convolutional kernel $\mathcal{K}$ such that $u = \mathcal{K} \odot z_0$ (as explained in Section 2). A time series of length T and dimension d passes through `CNet` to produce $u \in \mathbb{R}^{T \times D}$, increasing features from d to D ($D \geq d$ improves performance). The 1-D convolution provides robustness to local fluctuations without added complexity.
- `FMLP` Architecture. A multilayer perceptron (MLP) operating in the frequency domain. Since u can be seen as a linear transformation of y (4), we approximate the inversion with an MLP in the frequency domain, rather than in the time domain as done in [22,10] to enrich the feature representation of the time series and to capture informative frequencies (Fig. 3). The time series is transformed via FFT, resulting in its frequency domain representation, then processed by the MLP with a learnable complex weight, and converted back with IFFT. To preserve temporal structure [29], u is divided into M patches $\{\mathrm{S}_1, \ldots, \mathrm{S}_M\}$ of size m, each passed through linear layers to model dependencies, then flattened and followed by another linear layer to restore the sequence to its original length T, resulting in the desired input u. The `FMLP` can be expressed as:

$$u = \texttt{FMLP}(z_0) = \text{Patching}(\text{IFFT}(\text{MLP}(\text{FFT}(z_0) \times \text{weight}))).$$

For both inversion methods, the time series z_0 is first passed through a linear layer to produce y, matching u's dimension and yielding a residual $r \in \mathbb{R}^{T \times D}$ that captures anomaly information in each feature dimension.

Algorithm 1 Deep2DO

Input: time series $\mathbf{z_0}$, window length T, Dimension D, observer order s, and $g = -a\mathbf{1}_s$ with $a \in [0, 1]$

Re-parameterization
Define $\mathbf{G}_g$ and w with depth D
Randomly initialize $\mathbf{p}$ and $\mathbf{MM}$
Compute $\mathbf{H}, \mathbf{L}, v,$ and q using $\mathbf{p}$ with (6) and (7)
Set g, $\mathbf{p}$ and $\mathbf{MM}$ as learnable parameters
Return $(\mathbf{G}_g, \mathbf{H}, \mathbf{L}, q, v, g, w)$
Input Definition
Define u : $u = \texttt{CNet}_\Theta(z_0)$ or $u = \texttt{FMLP}_\Theta(z_0)$
Define y: $y = \texttt{Linear}_W(z_0)$
Set $z(0) = 0$
for $k = 0, 1, \cdots, T-1$ **do**
  $z(k+1) = \mathbf{G}z(k) + \mathbf{H}u(k) + \mathbf{L}y(k)$
  $r(k) = vy(k) - wz(k) - qu(k)$
end for
Output: z, r

Re-parametrization The re-parameterization decouples the system for the design of the DO matrices $(\mathbf{G}, \mathbf{H}, \mathbf{L}, q, v, g)$ and avoids solving (5) to implement a DO for AD. In the theoretical method [7], the DO matrices are designed based on a parity vector as follows: $\mathbf{L} = -\begin{bmatrix} p_0 \; p_1 \; \dots \; p_{s-1} \end{bmatrix}^T - g\, p_s$ and $\mathbf{G}$ parameterized as $\mathbf{G}_g = \begin{bmatrix} \mathbf{G}_0 \; g \end{bmatrix}$ where

$$(\mathbf{G}_0)_{ij} = \begin{cases} 1, & \text{if } i = j+1, \\ 0, & \text{otherwise,} \end{cases} \quad i = 1, \dots, s, \; j = 1, \dots, s-1, \tag{6}$$

and $g \in \mathbb{R}^{s\times 1}$ is a user-defined column vector that yields the system stable. Then $\mathbf{H}$, q, v and w are obtained as follows: $v = p_s$, $w = \begin{bmatrix} 0 \cdots 0 \; 1 \end{bmatrix}$, and

$$\begin{bmatrix} \mathbf{H} \\ q \end{bmatrix} = \begin{bmatrix} p_0 + g_1 p_s & p_1 & \cdots & p_{s-1} & p_s \\ p_1 + g_2 p_s & p_2 & \cdots & p_s & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{s-1} + g_s p_s & p_s & \cdots & 0 & 0 \\ p_s & 0 & \cdots & 0 & 0 \end{bmatrix} \times \mathbf{MM} \quad \text{with} \quad \mathbf{MM} = \begin{bmatrix} \overline{\mathbf{D}} \\ \overline{\mathbf{CB}} \\ \overline{\mathbf{CAB}} \\ \vdots \\ \overline{\mathbf{CA}}^{s-2}\overline{\mathbf{B}} \\ \overline{\mathbf{CA}}^{s-1}\overline{\mathbf{B}} \end{bmatrix}. \tag{7}$$

We maintain this design as in (6) and (7), while the values of $\mathbf{p}$ and the matrix $\mathbf{MM}$ are randomly initialized and $g = -a\mathbf{1}_s$ with $a \in [0, 1]$ allowing to promote system stability. The DO matrices are instantiated with D parallel depth during the re-parameterization resulting in $\mathbf{G} \in \mathbb{R}^{s\times s\times D}$ and $w \in \mathbb{R}^{s\times D}$. For $\mathbf{H}$ and $\mathbf{L}$, we add a channel dimension to support Einstein summation, yielding shapes $\mathbf{H}, \mathbf{L} \in \mathbb{R}^{s\times 1\times D}$ and $v, q \in \mathbb{R}^{1\times D}$. During training, these matrices are learned by backpropagation. As shown in Algorithm 1, once the re-parametrization is

executed, the Deep2DO can be implemented in a recurrent way by mapping the defined inputs $(u, y) \in \mathbb{R}^{T\times D}$ to a state observer $z \in \mathbb{R}^{T\times s\times D}$ and a residual $r \in \mathbb{R}^{T\times D}$.

3.4 Stacked Deep2DO

We propose a stacked version of Deep2DO as $(\text{Deep2DO}^{(i)})_{i=1}^{N}$. Each Deep2DO layer takes (u_i, y_i) as input and outputs an embedding $z_i \in \mathbb{R}^{T\times s_i}$ where anomaly-relevant information becomes increasingly salient and a residual $r_i \in \mathbb{R}^{T\times D_i}$ used to define the final residual $\overline{r}$.

In stacked Deep2DO, the time series z_0 is first processed by the input definition module to generate (u_1, y_1). For $i \geq 1$, the embedding z_i is reprocessed to form (u_{i+1}, y_{i+1}), which serves as input to $\text{Deep2DO}^{(i+1)}$. The final embedding is obtained recursively as

$$z_N = \text{Deep2DO}^{(N)}\big(\text{Deep2DO}^{(N-1)}(\cdots(\text{Deep2DO}^{(1)}(z_0)))\big). \tag{8}$$

Stacking enables progressive representation learning, where hierarchical refinement reduces complexity while preserving essential anomaly characteristics.

Final Residual. Each $\text{Deep2DO}^{(i)}$ with input (u_i, y_i) produces a residual $r_i \in \mathbb{R}^{T\times D_i}$ as in (3). Since residuals can be positive or negative [6], we take their absolute value to preserve anomaly magnitude and ensure compatibility with positive-valued metrics. The D_i dimensions of r_i are then aggregated via 1-D average pooling to aggregate anomaly information from all dimensions to obtain $\overline{r_i} \in \mathbb{R}^{T\times 1}$.

$$\overline{r_i} = \text{AdaptiveAvgPool1d}_1\big(|r_i|\big), \tag{9}$$

The N pooled residuals are summed to obtain the final residual $\overline{r}$ and to define the anomaly label $l(z_0)$ as follows:

$$\overline{r} = \sum_{i=1}^{N}(\overline{r_i}), \tag{10}$$

$$l(z_0(k)) = \begin{cases} 1 & \text{if } \overline{r}(k) > \tau \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

The stacked Deep2DO is trained on anomaly-free data z_0, where the mean squared error of $\overline{r}$ is used as the loss function and minimized accordingly. During inference, $\overline{r}$ serves as the anomaly score, which is thresholded by τ as in (11).

4 Experiments

4.1 Experimental Settings

Datasets. We train and evaluate our model on a diverse set of datasets to ensure a comprehensive evaluation across various scenarios. This includes anomalies ranging from point to sequence anomalies, in univariate and multivariate data. The datasets consist of real-world datasets and benchmarks. The real datasets consist of 5 datasets split as in [20]: MSL [40], SMAP [40], SWaT [27], PSM [1], SWAN [19]. The two benchmarks are: TSBAD [24], which comprises 180

Table 1: Models performance on the 5 real-world datasets. The best ones are in **bold**, and the second-best ones are underlined.

	MSL		SMAP		SWaT		PSM		SWAN	
	VUS-ROC	VUS-PR	VUS-ROC	VUS-PR	VUS-ROC	VUS-PR	VUS-ROC	VUS-PR	VUS-ROC	VUS-PR
OCSVM	0.5798	0.1753	0.4185	0.1134	0.5903	0.4396	0.5993	0.4252	0.9088	0.9004
PCA	0.6108	0.1889	0.4090	0.1144	0.6149	0.4459	0.6331	0.4706	0.9290	0.9123
iForest	0.5638	0.1631	0.4960	0.1315	0.3677	0.1011	0.6009	0.3964	0.8835	0.8793
LODA	0.5732	0.1689	0.3973	0.1017	0.6358	0.3531	0.6089	0.4423	09170	09107
HBOS	0.6265	0.1790	0.5620	0.1388	0.7084	0.4602	0.7056	0.5061	0.9056	0.8894
LOF	0.6081	0.1715	0.5673	0.1409	0.6667	0.4187	0.6628	0.4615	0.9095	0.9007
AE	0.6047	0.1890	0.4687	0.1366	0.5903	0.4144	0.6339	0.4490	0.6982	0.0201
LSTM	0.6163	0.1681	0.5329	0.1399	0.5482	0.2200	0.5571	0.459	0.9082	0.8862
Omni	0.5409	0.1973	0.4743	0.1239	0.6187	0.4475	0.6340	0.4472	0.9041	0.9022
GPT4TS	0.7697	0.2769	0.5449	0.1289	0.2537	0.0846	0.6466	0.4599	0.9340	0.8924
ModernTCN	0.7747	0.3010	0.5470	0.1395	0.2735	0.0941	0.6480	0.4668	0.9027	0.8962
MtsCID	0.4686	0.1181	0.4260	0.1177	0.5021	0.1283	0.5194	0.3296	0.8128	0.8375
TimesNet	0.7880	0.2731	0.5495	0.1352	0.2974	0.1158	0.6344	0.4373	0.9515	0.9160
CrossAD	0.8091	0.3144	0.5779	0.1433	0.7865	0.4767	0.7302	0.5596	0.9499	0.9171
KanAD	0.6444	0.1653	0.5563	0.1507	0.7841	0.1382	0.6844	0.4744	0.9368	0.8928
Deep2DO	**0.8370**	**0.3522**	**0.5801**	**0.1551**	**0.8737**	**0.7134**	**0.7908**	**0.5650**	**0.9666**	**0.9463**

multivariate and 350 univariate datasets, and UCR [44], which contains 250 univariate time series sub-datasets, each with a single anomaly segment.

Baseline. For the real-world datasets, we compare our model against 15 state-of-the-art models. For the TSBAD benchmark, we report only the most recent and top performing models due to space constraints. The baselines range from statistical models, including OCSVM [35], PCA [18], IForest [23], HBOS [13], and LOF [5], to neural network models, including AutoEncoder (AE) [33], LSTM, ModernTCN [9], TimesNet [43], CrossAD [20], KanAD [47], MtsCID [45], Omni-Anomaly (Omni) [40], TransformerAD (TranAD) [37], LSTMAD [26], CNN [41], Timer [25] and DADA [36]. We also include foundation models, with few-shot Moment (MomentFM) [14], zero-shot Moment (MomentZS) [14], and Chronos [2].

Metrics. We evaluate performance using classical metrics, including accuracy (Acc), Affiliation-F1 (Aff-F1), F1 score (F1), etc. However, these metrics exhibit certain limitations, and more robust alternatives have been proposed such as volume under the surface of precision-recall curve (VUS-PR) and of receiver operator characteristic curve (VUS-ROC) [4]. We employ these metrics predominantly for validation, enabling a rigorous evaluation of point and segment AD.

Implementation Details. Unless otherwise stated, the same hyperparameters are used for all experiments. We set $a = 0.1$, patch size $m = 1$, and $s \in \{5, 7\}$, with $s = 7$ for SWAT, SMAP, and UCR, and $s = 5$ otherwise. We use $D = 2d$ for `CNet` and $D = 60$ for `FMLP`. All models are trained with AdamW ($lr = 10^{-3}$). The window size T is specific to each dataset, while benchmark experiments use the same T across datasets. We set $N = 1$, except for the univariate MSL dataset where we use $N = 3$ with stacked dimensions $[10, 20, 5]$. `CNet` is used as the inversion block except for SWaT, where `FMLP` is applied. We use SPOT [39] to determines the anomaly threshold as in [20]. Experiments are implemented in PyTorch on a NVIDIA GRID A100D GPU (40GB VRAM).

Table 2: Detailed comparison on the real datasets with different metrics.

Dataset	Method	Acc	F1	Aff-F1	AUC-ROC	AUC-PR	R-A-R	R-A-P	VUS-ROC	VUS-PR
MSL	ModernTCN	0.7123	0.3254	0.7717	0.7602	0.2424	0.8040	0.3199	0.7747	0.3010
	TimesNet	**0.7491**	0.2996	0.7727	0.7395	0.2237	0.8019	0.3153	0.7880	0.2731
	KanAD	0.5647	0.2402	**0.7876**	0.6056	0.1487	0.6620	0.1777	0.6444	0.1653
	CrossAD	0.6864	**0.3378**	0.7740	**0.7812**	**0.2892**	0.8092	0.3466	0.8091	0.3144
	Deep2DO	0.5819	0.2869	0.6777	0.7376	0.2269	**0.8115**	**0.4234**	**0.8370**	**0.3522**
PSM	ModernTCN	0.3016	0.4345	0.7959	0.5846	0.3843	0.6550	0.4748	0.6480	0.4668
	TimesNet	0.2773	0.4342	0.7970	0.5755	0.3701	0.6617	0.4827	0.6344	0.4373
	KanAD	0.5849	0.4562	0.7409	0.6398	0.4323	0.6914	0.4940	0.6844	0.4744
	CrossAD	0.6847	0.4974	**0.8603**	0.6810	0.4906	0.7564	**0.6032**	0.7302	0.5596
	Deep2DO	**0.7172**	**0.5678**	0.8269	**0.7659**	**0.5260**	**0.8005**	0.5935	**0.7908**	**0.5650**
SWAN	ModernTCN	0.3256	0.4913	0.7109	0.5099	0.4411	0.9164	0.9138	0.9027	0.8962
	TimesNet	0.3256	0.4913	0.7109	0.5668	0.4841	0.9301	0.9261	0.9515	0.9160
	KanAD	0.7179	**0.6197**	0.7110	**0.7525**	0.5928	0.9176	0.9126	0.9368	0.8928
	CrossAD	0.3252	0.4905	0.7104	0.6010	0.5023	0.9408	0.9407	0.9499	0.9171
	Deep2DO	**0.7186**	0.5974	**0.7600**	0.6473	**0.6562**	**0.9589**	**0.9609**	**0.9666**	**0.9463**
SWAT	ModernTCN	0.1214	0.2165	0.7113	0.2492	0.0964	0.2892	0.1179	0.2735	0.0941
	TimesNet	0.9251	0.2051	0.7446	0.5852	0.1373	0.6683	0.2440	0.2974	0.1158
	KanAD	0.6782	0.2273	**0.8000**	0.7660	0.1229	0.7866	0.1422	0.7841	0.1382
	CrossAD	0.9162	0.6422	0.7605	0.8463	0.5831	0.7913	0.4848	0.7864	0.4767
	Deep2DO	**0.9732**	**0.7411**	0.7165	**0.8510**	**0.6615**	**0.8774**	**0.7342**	**0.8737**	**0.7134**

Table 3: Performance comparison on the univariate and multivariate TSBAD. The best results are shown in **bold**, while the second-best results are underlined. The "–" indicates that the model is not implemented for multivariate scenarios.

	Multivariate				Univariate			
Model	AUC-ROC	VUS-PR	VUS-ROC	Aff-F1	AUC-ROC	VUS-PR	VUS-ROC	Aff-F1
Deep2DO	**0.75**	**0.35**	0.75	**0.88**	0.77	**0.45**	**0.84**	**0.91**
CrossAD	0.74	0.33	**0.77**	0.86	**0.78**	0.45	0.84	0.89
CNN	0.73	0.31	0.76	0.87	0.71	0.34	0.79	0.88
Omnianomaly	0.65	0.31	0.69	0.81	0.65	0.29	0.72	0.83
PCA (Sub-PCA)	0.70	0.31	0.74	0.85	0.71	0.42	0.76	0.85
AutoEncoder	0.67	0.30	0.69	0.80	0.63	0.26	0.69	0.82
OCSVM (Sub-OCSVM)	0.61	0.26	0.67	0.80	0.65	0.23	0.73	0.79
TimesNet	0.56	0.19	0.64	0.82	0.18	0.61	0.26	0.86
TranAD	0.67	0.30	0.69	0.80	0.57	0.26	0.68	0.83
IForest	0.66	0.20	0.69	0.80	0.71	0.30	0.78	0.84
Chronos	-	-	-	-	0.66	0.27	0.73	0.88
Moment (FT)	-	-	-	-	0.69	0.39	0.76	0.86
Moment (ZS)	-	-	-	-	0.68	0.38	0.75	0.86
LOF	0.53	0.14	0.60	0.76	0.58	0.17	0.68	0.79

4.2 Results

Real-world Datasets. We evaluate our model against 15 baselines on five real-world datasets. Table 1 shows that our model outperforms the baselines across all five datasets in terms of the most reliable metrics, VUS-ROC and VUS-PR. A detailed comparison further supports these results, as given in Table 2, demonstrating that our model excels on five additional metrics: Accuracy, F1, AUC-PR, R-A-R, and R-A-P, where it is consistently ranked first or second.

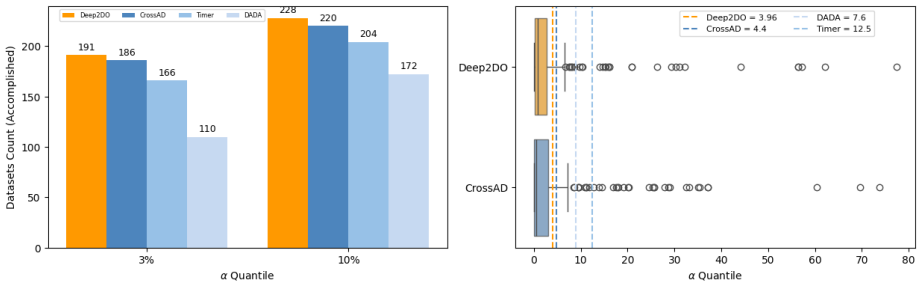

Fig. 4: Evaluation on the UCR Benchmark. Left: number of datasets where models detect anomalies at 3% and 10% quantiles. Right: comparison of the α quantile distribution, showing the mean. Lower values indicate earlier AD on average.

TSBAD Benchmark. A comparison of our model on the univariate and multivariate TSBAD benchmark against the 40 models of the benchmark also demonstrates in Table 3 strong performance relative to the baselines across the metrics AUC-ROC, VUS-PR, VUS-ROC, and Aff-F1. Our model consistently ranks first for VUS-PR and Aff-F1, and achieves either the best or second-best performance for the other metrics, with only a slight difference of $\approx 1\%$ in AUC-ROC.
UCR Benchmark. We follow CrossAD, Timer, and DADA approach to evaluate the UCR benchmark using the α-Quantile dataset count. Time steps are ranked by anomaly score, and detection is considered successful if the time step at a given α-Quantile falls within the labeled anomaly interval in the test set. The results in Fig. 4 show that our method ranks first at $\alpha = 3\%$ and 10%, detecting anomalies in more datasets and demonstrating strong early-detection performance. It also achieves a lower mean α-Quantile (3.96 vs. 4.4), indicating earlier detection on average. The boxplots in Fig. 4 further support these results, showing more gradually separated outliers and a more stable median and interquartile range. This indicates a more concentrated score distribution, making our method more robust and reproducible for α-Quantile-based AD.

4.3 Showcases

In Fig. 5, we present a visualization analysis of the Deep2DO variables and their derived variables. A single layer ($N = 1$) is trained on a 1D sinusoidal synthetic time series with 4.9% global point anomalies, using $s = 3$, $T = 50$, and $a = 0.1$. With $D = 2d = 2$, the inversion block yields $u = (u_1, u_2) \in \mathbb{R}^{T\times 2}$ without fault contamination. The DO embeds the data into $\mathbb{R}^{T\times 3\times 2}$ with $z = (z_{\cdot,1}, z_{\cdot,2})$, where in $z_{3,2}$ the anomalies are amplified. A residual $r = (r_1, r_2) \in \mathbb{R}^{T\times 2}$ is also obtained. Its derived variables from (3), $wz_{\cdot,2}$ and $(vy_2 - qu_2)$, exhibit opposing wave patterns, resulting in near-zero residuals for normal points, while r_2 shows pronounced spikes around anomalies. Consequently, the anomaly score $\overline{r}$ remains low on normal data, achieving VUS-PR = 0.9928 and Aff-F1 = 0.9739. The results highlight similarities between the residuals and the embeddings. This

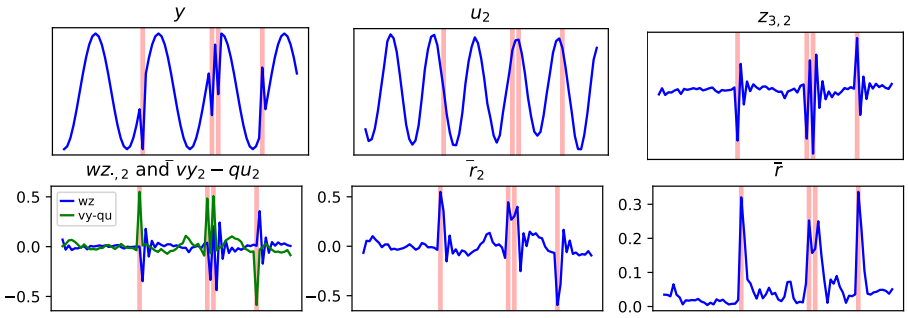

Fig. 5: Univariate synthetic global anomaly. Top left and center: (u, y) from the inversion block. Top right: one embedding dimension $z_{3,2}$. Bottom: DO components $wz_{\cdot,2}$, $vy_2 - qu_2$, and anomaly score $\overline{r}$. The model achieves VUS-PR = 0.9928 and Aff-F1 = 0.9739.

behavior is also consistent with the classical DO in dynamical systems [7]. This motivates stacking Deep2DO layers to detect anomalies in the embedding space.

4.4 Ablation Analysis

The ablation study is performed on the input definition process on SWaT and PSM datasets. In Fig. 6, we show the performance of Deep2DO with five different DL architectures to recover u: 1) `CNet` is our convolutional-based approach. 2) `FMLP` is our linear-based inversion approach. 3) w/o Fourier is `FMLP` with the Fourier transformation eliminated. 4) w/o Patching is `FMLP` with the patching block removed. 5) FNO [22] approximates the inversion by a linear transformation on both the frequency and time domains of the time series. The ablation study confirms the effectiveness of both inversion blocks. `FMLP` matches traditional convolutional blocks and FNO, achieving the best results on SWaT. In contrast, time-domain inversion (w/o Fourier) or removing the patching block reduces performance on both SWaT and PSM.

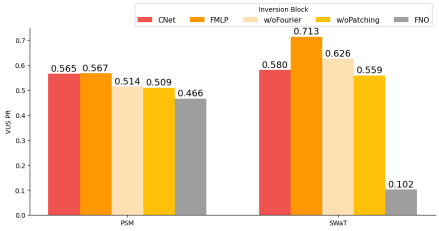

Fig. 6: VUS-PR metric changes with the ablation analysis.

4.5 Hyperparameter Analysis

We analyse Deep2DO's hyperparameter sensitivity in Fig. 7. The first three plots with $N = 1$ show that embedding size s and window length T have minimal effects, on the order of 10^{-2}. In contrast, the parameter a significantly impacts stability, with $a = 0.1$ performing best. The final plot presents the analysis for

Table 4: Efficiency study comparison results on PSM with 25 features

Model	Training Time (s)	Inference Time (s)	Total Params (M)	VUS PR
Ours (`CNet`)	78	0.74	0.40	<u>0.565</u>
Ours (`FMLP`)	361	2.97	0.22	**0.567**
CrossAD	927	0.36	1.32	0.559
TimesNet	144	23.92	4.69	0.437
ModernTCN	208	37.19	4.13	0.468

Table 5: Robustness study results over 5 runs

Dataset	VUS ROC	VUS PR
MSL	0.82 $\pm$ 0.008	0.32 $\pm$ 0.019
SWaT	0.87 $\pm$ 0.007	0.65 $\pm$ 0.081
SWAN	0.965 $\pm$ 0.002	0.94 $\pm$ 0.003

the stacked Deep2DO as N varies. We restrict the study to $N \in \{1, 2, 3\}$ to limit additional parameters for larger N. Fig. 7 (on the right) shows that increasing N is only beneficial when a single block is insufficient, improving performance on MSL for $N = 2$. Overall, the sensitivity of Deep2DO is moderate compared to many AD models, which typically require careful hyperparameter tuning.

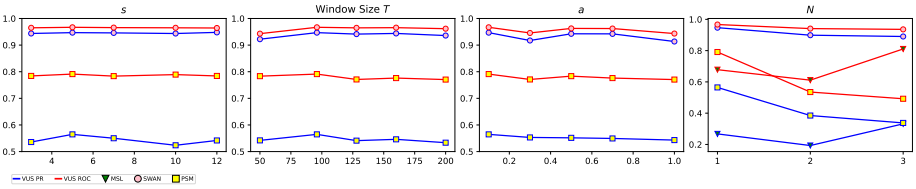

Fig. 7: Results of hyperparameters sensitivity analysis for the embedding size z, window size T, parameter a, and the number of stacked Deep2DO layers N

4.6 Efficiency Analysis

To evaluate efficiency, we report in Table 4, training time, inference time on the test set, total number of parameters, and performance of different DL models measured by the VUS-PR metric on the PSM dataset ($d = 25$). We assess the efficiency of Deep2DO by comparing it to different type of AD models (TimesNet, CrossAD, ModernTCN) using their best hyperparameters. Deep2DO achieves the highest VUS-PR with the smallest model size, while the `CNet` variant trains three times faster, offering an excellent performance-efficiency trade-off.

4.7 Robustness

We conduct a robustness analysis by training our model over 5 runs. The results are reported as the mean and standard deviation across these runs, providing insight into the stability and reliability of the model under varying initial conditions. The analysis is done under different scenarios: the stacked configuration

on the MSL dataset, the inversion with `CNet` on the SWaN dataset, and the inversion with `FMLP` on the SWaT dataset. The results in Table 5 show that our model with `CNet` inversion on the SWaN dataset is robust, exhibiting small variations on the order of 10^{-3}, with mean VUS-ROC and VUS-PR performance exceeding the state of the art. Similarly, the model with the `FMLP` and stacked Deep2DO architecture consistently achieves mean performance above the state of the art on the SWaT and MSL dataset, with variations on the order of 10^{-3} for VUS-ROC and showing slightly more variation for VUS in the order of 10^{-2}.

5 Conclusion

This paper introduced Deep2DO, the first DL-based framework that decouples DO from system modeling for time series AD. Deep2DO relied on two processes, input definition and re-parameterization, to design and define an optimal DO for AD. `CNet` and `FMLP` demonstrated strong representation learning capability compared to traditional architectures. Experiments on five real-world datasets and two large-scale benchmarks (536+ datasets) showed earlier detection on average and superior performance over more than 40 state-of-the-art models. The stacked version provided scalable and adaptable AD, with low sensitivity to key hyperparameters, particularly the embedding dimension. Benchmark evaluations under unified training settings further demonstrated strong generalization across anomaly types, paving the way for future zero-shot designs in time series AD.

References

1. Abdulaal, A., Liu, Z., Lancewicki, T.: Practical approach to asynchronous multivariate time series anomaly detection and localization. In: Proceedings of the 27th ACM SIGKDD. pp. 2485–2494 (2021)
2. Ansari, A.F., Stella, L., Turkmen, C., Zhang, X., Mercado, P., Shen, H., Shchur, O., Rangapuram, S.S., Arango, S.P., Kapoor, S., et al.: Chronos: Learning the language of time series. Transactions on Machine Learning Research **2024** (2024)
3. Bao, J., Sun, H., Deng, H., He, Y., Zhang, Z., Li, X.: BMAD: Benchmarks for medical anomaly detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4042–4053 (2024)
4. Boniol, P., Krishna, A.K., Bruel, M., Liu, Q., Huang, M., Palpanas, T., Tsay, R.S., Elmore, A., Franklin, M.J., Paparrizos, J.: Vus: effective and efficient accuracy measures for time-series anomaly detection. The VLDB Journal **34**(3) (Mar 2025)
5. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD international conference on Management of data. pp. 93–104 (2000)
6. Ding, S.X., Yin, S., Wang, Y., Wang, Y., Yang, Y., Ni, B.: Data-driven design of observers and its applications. IFAC Proceedings Volumes **44**(1), 11441–11446 (2011), 18th IFAC World Congress
7. Ding, S.X.: Model-Based Fault Diagnosis Techniques: Design Schemes, Algorithms, and Tools, Advances in Industrial Control, vol. 381. Springer, Berlin, Heidelberg (2008)

8. Ding, Y., Ceglarek, D., Shi, J.: Fault diagnosis of multistage manufacturing processes by using state space approach. Journal of Manufacturing Science and Engineering **Vol.124**(No.2), 313–322 (2002)
9. Donghao, L., Xue, W.: ModernTCN: A modern pure convolution structure for general time series analysis. In: The Twelfth ICLR (2024)
10. Eldele, E., Ragab, M., Chen, Z., Wu, M., Li, X.: Tslanet: rethinking transformers for time series representation learning. In: Proceedings of the 41st ICML. JMLR.org (2024)
11. Esling, P., Agon, C.: Time-series data mining. ACM Comput. Surv. **45**(1) (Dec 2012)
12. Gertler, J.: Fault Detection and Diagnosis in Engineering Systems. Marcel Dekker (1998)
13. Goldstein, M., Dengel, A.: Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. KI-2012: poster and demo track **1**, 59–63 (2012)
14. Goswami, M., Szafer, K., Choudhry, A., Cai, Y., Li, S., Dubrawski, A.: Moment: a family of open time-series foundation models. In: Proceedings of the 41st ICML (2024)
15. Gu, A., Dao, T.: Mamba: Linear-time sequence modeling with selective state spaces. In: First Conference on Language Modeling (2024)
16. Gu, A., Goel, K., Re, C.: Efficiently modeling long sequences with structured state spaces. In: ICLR (2022)
17. He, H., Bai, Y., Zhang, J., He, Q., Chen, H., Gan, Z., Wang, C., Li, X., Tian, G., Xie, L.: MambaAD: Exploring state space models for multi-class unsupervised anomaly detection. Advances in NeurIPS **37**, 71162–71187 (2024)
18. Jolliffe, I., et al.: Principal component analysis: a review and recent developments. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **374**(2065), 20150202 (Apr 2016)
19. Lai, K.H., Zha, D., Xu, J., Zhao, Y., Wang, G., Hu, X.: Revisiting time series outlier detection: Definitions and benchmarks. In: Thirty-fifth Conference on NeurIPS Datasets and Benchmarks Track (Round 1) (2021)
20. Li, B., Shentu, Q., Shu, Y., Zhang, H., Li, M., Jin, N., Yang, B., Guo, C.: CrossAD: Time series anomaly detection with cross-scale associations and cross-window modeling. In: The Thirty-ninth Annual Conference on NeurIPS (2025)
21. Li, Y., Chen, W., Chen, B., Wang, D., Tian, L., Zhou, M.: Prototype-oriented unsupervised anomaly detection for multivariate time series. In: Proceedings of the 40th International Conference on Machine Learning. JMLR.org (2023)
22. Li, Z., Kovachki, N.B., Azizzadenesheli, K., liu, B., Bhattacharya, K., Stuart, A., Anandkumar, A.: Fourier neural operator for parametric partial differential equations. In: ICLR (2021)
23. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining. pp. 413–422. IEEE (2008)
24. Liu, Q., Paparrizos, J.: The elephant in the room: Towards a reliable time-series anomaly detection benchmark. In: The Thirty-eight Conference on NeurIPS Datasets and Benchmarks Track (2024)
25. Liu, Y., Zhang, H., Li, C., Huang, X., Wang, J., Long, M.: Timer: Transformers for time series analysis at scale. ArXiv **abs/2402.02368** (2024)
26. Malhotra, P., Vig, L., Shroff, G., Agarwal, P., et al.: Long short term memory networks for anomaly detection in time series. In: ESANN. vol. 2015, p. 89 (2015)
27. Mathur, A.P., Tippenhauer, N.O.: Swat: A water treatment testbed for research and training on ics security. In: 2016 international workshop on cyber-physical systems for smart water networks (CySWater). pp. 31–36. IEEE (2016)

28. Mishra, V.K., Iannelli, A., Bajcinca, N.: A data-driven approach to system invertibility and input reconstruction. In: 62nd IEEE Conference on Decision and Control (CDC). pp. 671–676 (2023)
29. Nie, Y., Nguyen, N.H., Sinthong, P., Kalagnanam, J.: A time series is worth 64 words: Long-term forecasting with transformers. In: The Eleventh ICLR (2023)
30. Ogata, K.: Modern Control Engineering. Pearson, Upper Saddle River., 5th edn. (2010)
31. Pushkov, S.G.: Inversion of linear systems on the basis of state space realization. Journal of Computer and Systems Sciences International **57**(1), 7–17 (2018)
32. Sain, M.K., Massey, J.L.: Invertibility of linear time-invariant dynamical systems. IEEE Transactions on Automatic Control **AC-14**(2), 141–149 (1969)
33. Sakurada, M., Yairi, T.: Anomaly detection using autoencoders with nonlinear dimensionality reduction. In: MLSDA 2014. p. 4–11. New York, NY, USA (2014)
34. Schmidl, S., Wenig, P., Papenbrock, T.: Anomaly detection in time series: a comprehensive evaluation. Proc. VLDB Endow. **15**(9), 1779–1797 (May 2022)
35. Schölkopf, B., Williamson, R.C., Smola, A., Shawe-Taylor, J., Platt, J.: Support vector method for novelty detection. vol. 12. MIT Press (1999)
36. Shentu, Q., Li, B., Zhao, K., Shu, Y., Rao, Z., Pan, L., Yang, B., Guo, C.: Towards a general time series anomaly detector with adaptive bottlenecks and dual adversarial decoders. In: ICLR (2025)
37. Shreshth, T., et al.: Tranad: deep transformer networks for anomaly detection in multivariate time series data. Proc. VLDB Endow. **15**(6), 1201–1214 (Feb 2022)
38. Siddiqui, M.A., Stokes, J.W., Seifert, C., Argyle, E., McCann, R., Neil, J., Carroll, J.: Detecting cyber attacks using anomaly detection with explanations and expert feedback. In: IEEE ICASSP. pp. 2872–2876 (2019)
39. Siffer, A., Fouque, P.A., Termier, A., Largouet, C.: Anomaly detection in streams with extreme value theory. In: Proceedings of the 23rd ACM international conference on knowledge discovery and data mining. pp. 1067–1075 (2017)
40. Su, Y., Zhao, Y., Niu, C., Liu, R., Sun, W., Pei, D.: Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In: Proceedings of the 25th ACM SIGKDD. p. 2828–2837. KDD '19 (2019)
41. Tuli, S., Casale, G., Jennings, N.R.: Deepant: A deep learning approach for unsupervised anomaly detection in time series. IEEE Access **7**, 1991–2005 (2018)
42. Wang, Z., Kong, F., Feng, S., Wang, M., Yang, X., Zhao, H., Wang, D., Zhang, Y.: Is mamba effective for time series forecasting? Neurocomputing **619** (2024)
43. Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., Long, M.: Timesnet: Temporal 2d-variation modeling for general time series analysis. In: ICLR (2022)
44. Wu, R., Keogh, E.J.: Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. IEEE TKDE (2023)
45. Xie, Y., Zhang, H., Babar, M.A.: Multivariate time series anomaly detection by capturing coarse-grained intra-and inter-variate dependencies. In: Proceedings of the ACM on Web Conference 2025. pp. 697–705 (2025)
46. Zhang, P., Ding, S.X.: Disturbance decoupling in fault detection of linear periodic systems. Automatica **43**(8), 1410–1417 (2007)
47. Zhou, Q., Pei, C., Sun, F., HanJing, Gao, Z., Zhang, H., Xie, G., Pei, D., li, J.: KAN-AD: Time series anomaly detection with kolmogorov–arnold networks. In: Forty-second ICML (2025)
48. Zhou, T., Niu, P., Wang, X., Sun, L., Jin, R.: One fits all: Power general time series analysis by pretrained lm. In: Advances in NeurIPS. vol. 36, pp. 43322–43355 (2023)