arXiv:2306.15489v3 [cs.AI] 13 Oct 2023

Precursor-of-Anomaly Detection for Irregular Time Series

Sheo Yon Jhin
Yonsei University
Seoul, South Korea
sheoyonj@yonsei.ac.kr

Jaehoon Lee*
LG AI Research & Yonsei University
Seoul, South Korea
jaehoon.lee@lgresearch.ai

Noseong Park
Yonsei University
Seoul, South Korea
noseong@yonsei.ac.kr

ABSTRACT

Anomaly detection is an important field that aims to identify unexpected patterns or data points, and it is closely related to many real-world problems, particularly to applications in finance, manufacturing, cyber security, and so on. While anomaly detection has been studied extensively in various fields, detecting future anomalies before they occur remains an unexplored territory. In this paper, we present a novel type of anomaly detection, called ***P**recursor-of-**A**nomaly* (PoA) detection. Unlike conventional anomaly detection, which focuses on determining whether a given time series observation is an anomaly or not, PoA detection aims to detect future anomalies before they happen. To solve both problems at the same time, we present a neural controlled differential equation-based neural network and its multi-task learning algorithm. We conduct experiments using 17 baselines and 3 datasets, including regular and irregular time series, and demonstrate that our presented method outperforms the baselines in almost all cases. Our ablation studies also indicate that the multitasking training method significantly enhances the overall performance for both anomaly and PoA detection.

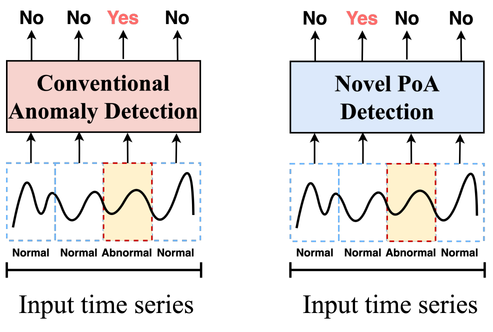
CCS CONCEPTS

• **Computing methodologies** → *Machine learning algorithms*; **Artificial intelligence**.

KEYWORDS

time-series, anomaly detection, multi-task learning

ACM Reference Format:
Sheo Yon Jhin, Jaehoon Lee, and Noseong Park. 2023. Precursor-of-Anomaly Detection for Irregular Time Series. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3580305.3599469



(a) Anomaly detection (b) PoA detection

Figure 1: Comparison between the conventional anomaly detection and our proposed the precursor-of-anomaly (PoA) detection. In PoA, we predict whether the next window will contain any abnormal observation before it happens, which is much more challenging than the anomaly detection.

1 INTRODUCTION

Anomaly detection is an important and difficult task that has been extensively studied for many real-world applications [12, 16, 44, 51]. The goal of anomaly detection is to find unexpected or unusual data points and/or trends that may indicate errors, frauds, or other abnormal situations requiring further investigations.

Novel task definition: Among many such studies, one of the most popular setting is *multivariate time series anomaly detection* since many real-world applications deal with time series, ranging from natural sciences, finance, cyber security, and so on. Irregular time series anomaly detection [9, 27, 45] is of utmost importance since time series data is frequently irregular. Many existing time series anomaly detection designed for regular time series show sub-optimal outcomes when being applied to irregular time series. Irregular time series typically have complicated structures with uneven inter-arrival times.

In addition, it is also important to forecast whether there will be an anomaly in the future given a time series input, which we call ***P**recursor-of-**A**nomaly* (PoA) detection. The precursor-of-anomaly detection refers to the process of identifying current patterns/signs that may indicate upcoming abnormal events. ? The goal is to detect these precursors before actual anomalies occur in order to take preventive actions (cf. Fig. 1). The precursor-of-anomaly detection can be applied to various fields such as finance, medical care, and geophysics. For example, in finance, the precursor-of-anomaly detection can be used to identify unusual patterns in stock prices that may indicate a potential market collapse, and in geophysics, the precursor-of-anomaly detection can detect unusual geometric activities that can indicate earthquakes. Despite of its practical usefulness, the precursor-of-anomaly detection has been overlooked for a while due to its challenging nature except for a

*This work was done when he was at Yonsei university.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD '23, August 6–10, 2023, Long Beach, CA, USA.
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599469

handful of work to detect earthquakes based on manual statistical analyses [6, 13, 37].

To this end, we mainly focus on the following two tasks for multivariate and irregular time series: i) detecting whether the input time series sample has anomaly or not, i.e., detecting an anomaly, and ii) detecting whether anomaly will happen after the input time series sample, i.e., detecting the precursor-of-anomaly. In particular, we are the first detecting the precursor-of-anomaly for general domains. Moreover, we solve the two related problems at the same time with a single framework.

Novel method design: Our work is distinctive from existing work not only in its task definition but also in its deep learning-based method design. Recent anomaly detection methods can be divided into three main categories: i) clustering-based ii) density estimation-based iii) reconstruction-based methods. Clustering-based methods use historical data to identify and measure the distance between normal and abnormal clusters but can struggle to detect anomalies in complex data where distinct abnormal patterns are unknown [1]. Density estimation-based methods estimate a density function to detect anomalies. These methods assume that abnormal samples lie in a low density region. However, there is a limitation in data where the difference between low-probability normal and abnormal samples is not clear [54]. On the other hand, reconstruction-based methods aim to identify abnormal patterns by learning low-dimensional representations of data and detecting anomalies based on reconstruction errors. However, these methods may have limitations in modeling dependencies and temporal dependencies in time series, making them unsuitable for detecting certain types of anomalies [2, 50]. To detect anomalies that may not be apparent in raw time-series, there are several advanced methods [22, 31].

In this paper, we propose i) a unified framework of detecting the anomaly and precursor-of-anomaly based on neural controlled differential equations (NCDEs) and ii) its multi-task learning and knowledge distillation-based training algorithm with self-supervision. NCDEs are recently proposed differential equation-based *continuous-time recurrent neural networks* (RNNs), which have been widely used for irregular time series forecasting and classification. For the first time, we adopt NCDEs to time series (precursor of) anomaly detection. Our method, called (**P**recursor of) **A**nomaly **D**etection (PAD), is able to provide both the anomaly and the precursor-of-anomaly detection capabilities.

As shown in Fig. 3 there are two co-evolving NCDEs: i) the anomaly NCDE layer for the anomaly detection, ii) and the PoA NCDE layer for the precursor-of-anomaly detection. They are trained as a single unified framework under the multi-task training and knowledge distillation regime. Since our method is trained in a self-supervised manner, we resample the normal training dataset to create artificial anomalies (see Section 3.5). To train the PoA NCDE layer, we apply a knowledge distillation method where the anomaly NCDE layer becomes as a teacher and the PoA NCDE becomes a student. By conducting a training by knowledge distillation, the PoA NCDE can inherit the knowledge of the anomaly NCDE. In other words, the PoA NCDE layer, which reads a temporal window up to time i, mimics the inference of the anomaly NCDE which reads a window up to time $i+1$ and decide whether it is an anomaly, which is a possible PoA detection approach since the PoA NCDE layer sees only past information. The two different NCDEs co-evolve while interacting with each other. At the end, our proposed method is trained for all the two tasks. They also partially share trainable parameters, which is a popular the multi-task design.

We performed experiments with 17 baselines and 3 datasets. In addition to the experiments with full observations, we also test after dropping random observations in order to create challenging detection environments. In almost all cases, our proposed method, called *PAD*, shows the best detection ability. Our ablation studies prove that the multi-task learning method enhances our method's capability. We highlight the following contributions in this work:

(1) We are the first solving the anomaly and the PoA detection at the same time.
(2) We propose PAD which is an NCDE-based unified framework for the anomaly and the PoA detection.
(3) To this end, we also design a multi-task and knowledge distillation learning method.
(4) In addition, the above learning method is conducted in a self-supervised manner. Therefore, we also design an augmentation method to create artificial anomalies for our self-supervised training.
(5) We perform a comprehensive set of experiments, including various irregular time series experiments. Our PAD marks the best accuracy in almost all cases.
(6) Our code is available at this link [1], and we refer readers to Appendix for the information on reproducibility.

2 RELATED WORK

2.1 Anomaly Detection in Time Series

Anomaly detection in time series data has been a popular research area in the fields of statistics, machine learning and data mining. Over the years, various techniques have been proposed to identify anomaly patterns in time series data, including machine learning algorithms, and deep learning models.

Machine learning algorithms and deep learning-based anomaly detection methods in time series can be categorized into 4 methods including i) classical methods, ii) clustering-based methods, iii) density-estimation methods, iv) reconstruction-based methods.

Classical methods find unusual samples using traditional machine learning methods such as OCSVM and Isolation Forest [25, 43]. Clustering-based methods are a type of unsupervised machine learning method for detecting anomalies in time-series data such as Deep SVDD, ITAD, and THOC [35, 38, 39]. This method splits the data into multiple clusters based on the similarity between the normal data points and then identifies abnormal data points that do not belong to any clusters. However, clustering-based methods are not suitable for complex data to train. Density-estimation-based methods [7, 48] are a type of unsupervised machine learning method for detecting anomalies in time series data such as LOF. In density-estimation based anomaly detection, the time-series data is transformed into a feature space, and the probability density function of the normal data points is estimated using techniques such as kernel density estimation or gaussian mixture models. The data points are

[1]https://github.com/sheoyon-jhin/PAD

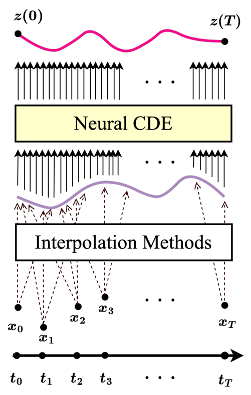


Figure 2: The architecture of NCDEs.

then ranked based on their density, and anomalies are identified as data points that have low density compared to the normal data points. However, this method may not perform well in cases where the data contains significant non-stationary patterns or where the anomalies are not well represented by the estimated density function. Additionally, this method can also be computationally expensive, as it requires estimating the density function for the entire data. Reconstruction-based methods are [4, 24, 32, 40] a representative methodology for detecting anomalies in time series data such as USAD [4]. In this method, a deep learning model reconstructs normal time-series data and then identifies abnormal time-series data with high reconstruction errors. Also, reconstruction-based methods can be effective in detecting anomalies in time-series data, especially when the anomalies are not well separated from the normal data points and the data contains complex patterns. The reconstruction-based methods are also robust to missing values and noisy data, as the reconstructed data can be used to fill in missing values and reduce the impact of noise. However, this method can be sensitive to the choice of reconstruction model, and may not perform well in cases where the anomalies are not well represented by the reconstruction model. Additionally, this method may be computationally expensive, as it requires training a complex model on the data.

In this paper, we mainly focus on self-supervised learning-based anomaly detection. Anomaly detection methods based on self-supervised learning have been extensively studied [8, 23, 36, 41]. Unlike the unsupervised-learning method, self-supervised learning learns using negative samples generated by applying data augmentation to the training data [8, 41]. Data augmentation can overcome the limitations of existing training data. In this paper, we train our method, PAD, on time series dataset generated by data augmentation.

2.2 Neural Controlled Differential Equations

Recently, differential equation-based deep learning models have been actively researched. Neural controlled differential equations (NCDEs) are a type of machine learning method for modeling and forecasting time-series data. NCDE combines the benefits of neural networks and differential equations to create a flexible and powerful model for time-series data. Unlike traditional time series models (e.g., RNNs), differential equation-based neural networks estimate the continuous dynamics of hidden vectors $\mathbf{z}(\mathbf{t})$ (i.e., $\frac{dz(t)}{dt}$). Neural ordinary differential equations (NODEs) use the following equations to model a hidden dynamics [10]:

$$\mathbf{z}(T) = \mathbf{z}(0) + \int_0^T f(\mathbf{z}(t), t; \theta_f)dt. \tag{1}$$

In contrast to NODEs, neural controlled differential equations (NCDEs) utilize the riemann–stieltjes integral [21]. Let $\{\mathbf{x}_i\}_{i=0}^N$ be time series observations whose length is N and $\{t_i\}_{i=0}^N$ be its observation time-points. NCDEs can be formulated as the following equations:

$$\mathbf{z}(T) = \mathbf{z}(0) + \int_0^T f(\mathbf{z}(t); \theta_f)dX(t), \tag{2}$$

$$= \mathbf{z}(0) + \int_0^T f(\mathbf{z}(t); \theta_f)\frac{dX(t)}{dt}dt, \tag{3}$$

where $X(t)$ is a continuous time series path interpolated from $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=0}^N$. With interpolation methods, we can obtain a continuous time series path from discrete observations $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=0}^N$. Typically, natural cubic spline [30] is used as the interpolation method — note that the adaptive step size solver works properly if the path $X(t)$ is twice continuously differentiable [21].

The main difference between NODEs and NCDEs is the existence of the continuous time series path $X(t)$. NCDEs process a hidden vector $\mathbf{z}(t)$ along the interpolated path $X(t)$. (cf. Fig. 2) In this regard, NCDEs can be regarded as a continuous analogue to RNNs. Because there exists $X(t)$, NCDEs can learn what NODEs cannot (See Theorem C.1 in [21]). Additionally, NCDEs have the ability to incorporate domain-specific knowledge into the model, by using expert features as inputs to the neural networks. This can improve the performance and robustness of the model, as well as provide a deeper understanding of the relationships between the variables in the data. Therefore, NCDEs enable us to model more complex patterns in continuous time series.

2.3 Knowledge Distillation

Knowledge distillation [19] is a technique used to transfer knowledge from large, complex models to smaller, simpler models. In the context of time series data, knowledge distillation can be used to transfer knowledge from complex deep learning models to smaller, more efficient models. The knowledge extraction process involves training complex models such as deep neural networks on large datasets of time series data. The complex model is then used to generate predictions for the same data set. These predictions are then used to train simpler models such as linear regression models or decision trees. In this paper, we use these predictions in precursor-of-anomaly detection. During the training process, simpler models are optimized to minimize the difference between their predictions and those of complex models. This allows simple models to learn patterns and relationships in time series data captured by complex models. Furthermore, knowledge distillation can also help improve the interpretability of the model. The simpler model can be easier to understand and analyze, and the knowledge transferred from the complex model can provide insights into the important features and relationships in the time-series data. The advantages of distilling knowledge from time series include faster and more efficient model training, reduced memory requirements, and improved interpretability of the model [5, 47].

2.4 Multi-task Learning

Multi-task learning (MTL) is a framework for learning multiple tasks jointly with shared parameters, rather than learning them independently [52]. By training multiple tasks simultaneously, MTL can take advantage of shared structures between tasks to improve the generalization performance of each individual task. Several different MTL architectures have been proposed in the literature, including hard parameter sharing, soft parameter sharing, and task-specific parameter sharing [34]. Hard parameter sharing involves using a shared set of parameters for all tasks, while soft parameter sharing allows for different parameter sets for each task, but encourages them to be similar. Task-specific parameter sharing has their own set of task-specific parameters for each task, and lower-level parameters are shared between tasks. This approach allows for greater task-specific adaptation while still leveraging shared information at lower levels.

One of the major challenges of MTL is balancing task-specific and shared representation. Using more shared representations across tasks yields better generalization and performance across all tasks, but at the expense of task-specific information. Therefore, the choice of MTL architecture and the amount of information shared between tasks depends on the specific problem and task relationship. In this paper, we train our model with a task-specific parameter sharing architecture considering shared information and task relationships. MTL is applied to various areas such as natural language processing, computer vision. In natural language processing, MTL has been used for tasks such as named entity recognition, part-of-speech tagging, and sentiment analysis [11]. In computer vision, MTL has been used for tasks such as object recognition, object detection, and semantic segmentation [26].

These architectures have demonstrated promising results on several benchmark datasets, and in this paper, we conduct research on time-series anomaly detection using MTL, showing excellent performance for all 17 baselines.

3 PROPOSED METHOD

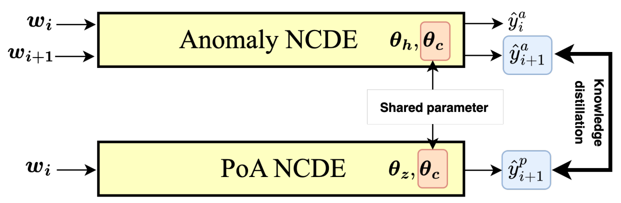
In this section, we describe the problem statement of our window-based time series anomaly and precursor-of-anomaly detection in detail and introduce the overall architecture of our proposed method, PAD, followed by our NCDE-based neural network design, its multi-task learning strategy and the knowledge distillation.

3.1 Problem Statement

!

In this section, we define the anomaly detection and the precursor-of-anomaly detection tasks in irregular time series settings — those problems in the regular time series setting can be similarly defined.

In our study, we focus on multivariate time series, defined as $\mathbf{x}_{0:T} = \{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_T\}$, where T is the time-length. An observation at time t, denoted $\mathbf{x}_t$, is an N-dimensional vector, i.e., $\mathbf{x}_t \in \mathbb{R}^N$. For irregular time-series, the time-interval between two consecutive



WiNDOWS

Figure 3: Overall Architecture

observations is not a constant — for regular time-series, the time-interval is fixed. We use a window-based approach where time series $\mathbf{x}_{0:T}$ is divided into non-overlapping windows, i.e., $\mathbf{w}_i = [\mathbf{x}_{t^i_0}, \mathbf{x}_{t^i_1}, ..., \mathbf{x}_{t^i_b}]$ where $t^i_j = i \times b + j - 1$ with a window size of b. There are $\lceil \frac{T}{b} \rceil$ windows in total for $\mathbf{x}_{0:T}$. Each window is individually taken as an input to our network for the anomaly or the precursor-of-anomaly detection.

!

Given an input window $\mathbf{w}_i$, for the anomaly detection, our neural network decides whether $\mathbf{w}_i$ contains abnormal observations or not, i.e., a binary classification of anomaly vs. normal. On the other hand, it is determined whether the very next window $\mathbf{w}_{i+1}$ is likely to contain abnormal observations for the precursor-of-anomaly detection, i.e., yet another binary classification.

3.2 Overall Workflow

Fig. 3 shows the detailed design of our method, PAD. The overall workflow is as follows:

(1) For self-supervised learning, we create augmented training data with data augmentation techniques (see Section 3.5).
(2) There are two co-evolving NCDE layers which produce the last hidden representations $\mathbf{h}(T)$ and $\mathbf{z}(T)$ in Eq. (4).
(3) In the training progress, the anomaly NCDE gets two inputs, w_i for the anomaly detection and w_{i+1} for the PoA detection.
(4) There are 2 output layers for the anomaly detection and the PoA detection, respectively. These two different tasks are integrated into a single training method via our shared parameter θ_c for multi-task learning.
(5) In the training progress, the anomaly NCDE creates the two outputs $\hat{y}^a_i$ and $\hat{y}^a_{i+1}$ for the knowledge distillation.

3.3 Neural Network Architecture based on Co-evolving NCDEs

We describe our proposed method based on dual co-evolving NCDEs: one for the anomaly detection and the other for the precursor-of-anomaly (PoA) detection. Given a discrete time series sample $\mathbf{x}_{1:T}$, we create a continuous path $X(t)$ using an interpolation method, which is a pre-processing step of NCDEs. After that, the following co-evolving NCDEs are used to derive the two hidden vectors $\mathbf{h}(T)$ and $\mathbf{z}(T)$:

$$\begin{aligned}\mathbf{h}(T) &= \mathbf{h}(0) + \int_0^T f(\mathbf{h}(t);\theta_f,\theta_c)\frac{dX(t)}{dt}dt,\\ \mathbf{z}(T) &= \mathbf{z}(0) + \int_0^T g(\mathbf{z}(t);\theta_g,\theta_c)\frac{dX(t)}{dt}dt,\end{aligned} \quad (4)$$

where the two functions (neural networks) f and g have their own parameter and shared parameter c

For instance, θ_f is specific to the function f whereas θ_c is a common parameter shared by f and g. The exact architectures of f and g are as follows:

$$f(\mathbf{h}(t);\theta_f,\theta_c) = \underbrace{\rho(\mathsf{FC}(\phi(\mathsf{FC}(\mathbf{h}(t)))))}_{\theta_f} + \underbrace{\rho(\mathsf{FC}(\phi(\mathsf{FC}(\mathbf{h}(t)))))}_{\theta_c}, \quad (5)$$

$$g(\mathbf{z}(t);\theta_g,\theta_c) = \underbrace{\rho(\mathsf{FC}(\phi(\mathsf{FC}(\mathbf{z}(t)))))}_{\theta_g} + \underbrace{\rho(\mathsf{FC}(\phi(\mathsf{FC}(\mathbf{z}(t)))))}_{\theta_c} \quad (6)$$

where ϕ means the rectified linear unit (ReLU) and ρ means the hyperbolic tangent.

Therefore, our proposed architecture falls into the category of *task-specific parameter sharing* of the multi-task learning paradigm. We perform the anomaly detection task and the PoA detection tasks with $\mathbf{h}(T)$ and $\mathbf{z}(T)$, respectively. We ensure that the two NCDEs co-evolve by using the shared parameters θ_c that allow them to influence each other during the training process. Although those two tasks' goals are ultimately different, those two tasks share common characteristics to some degree, i.e., capturing key patterns from time series. By controlling the sizes of θ_f, θ_g, and θ_c, we can control the degree of their harmonization. After that, we have the following two output layers:

$$\hat{y}_i^a = \sigma(\mathsf{FC}_{\theta_a}(\mathbf{h}(T))), \text{ for the anomaly detection,} \quad (7)$$

$$\hat{y}_i^p = \sigma(\mathsf{FC}_{\theta_p}(\mathbf{z}(T))), \text{ for the precursor-of-anomaly detection,} \quad (8)$$

where each fully-connected (FC) layer is defined for each task and σ is the sigmoid activation for binary classification.

When implementing the co-evolving NCDEs, we implement the following augmented state and solve the two initial value problems in Eq. (4) at the same time with an ODE solver:

$$\frac{d}{dt}\begin{bmatrix}\mathbf{h}(t)\\ \mathbf{z}(t)\end{bmatrix} = \begin{bmatrix} f(\mathbf{h}(t);\theta_f,\theta_c)\frac{dX(t)}{dt} \\ g(\mathbf{z}(t);\theta_g,\theta_c)\frac{dX(t)}{dt}\end{bmatrix}, \quad (9)$$

and

$$\begin{bmatrix}\mathbf{h}(0)\\ \mathbf{z}(0)\end{bmatrix} = \begin{bmatrix}\mathsf{FC}_{\theta_\mathbf{h}}(X(0))\\ \mathsf{FC}_{\theta_\mathbf{z}}(X(0))\end{bmatrix}. \quad (10)$$

Well-posedness of the problem: The well-posedness of the initial value problem of NCDEs was proved in previous work, such as [21, 28] under the condition of Lipschitz continuity, which means that the optimal form of the last hidden state at time T is uniquely defined given an training objective. Our method, PAD, also has this property, as almost all activation functions (e.g. ReLU, Leaky ReLU, Sigmoid, ArcTan, and Softsign) have a Lipschitz constant of 1 [21]. Other neural network layers, such as dropout, batch normalization, and pooling methods, also have explicit Lipschitz constant values. Therefore, Lipschitz continuity can be fulfilled in our case.

3.4 Training Algorithm

Loss function: We use the cross-entropy (CE) loss to train our model. L_{KD} and L_a mean the CE loss for the knowledge distillation and the anomaly detection, respectively:

$$L_{KD} = CE(\hat{y}_{i+1}^a, \hat{y}_{i+1}^p), \quad (11)$$

Algorithm 1: How to train PAD

```
Input: Training data $D_{train}$, Validating data $D_{val}$,
       Maximum iteration number $max\_iter$
/* $\theta_{others} = \{\theta_\mathbf{h}, \theta_\mathbf{z}, \theta_a, \theta_t, \theta_p\}$ */
1 Initialize $\theta_f$, $\theta_g$, $\theta_c$, and $\theta_{others}$;
2 $k \leftarrow 0$;
3 while $k < max\_iter$ do
4     Train $\theta_f$ and $\theta_{others}$ with $L_a$;
5     Train $\theta_g$ and $\theta_{others}$ with $L_{KD}$;
6     Train the common parameter $\theta_c$ with $L_a, L_{KD}$;
7     Validate and update the best parameters, $\theta_f^*$, $\theta_g^*$, $\theta_c^*$,
        $\theta_{others}^*$ with $D_{val}$;
8     $k \leftarrow k + 1$;
9 return $\theta_f^*$, $\theta_g^*$, $\theta_c^*$, and $\theta_{others}^*$;
```

where $\hat{y}_{i+1}^a$ denotes the anomaly NCDE model's output, $\hat{y}_{i+1}^p$ denotes the PoA NCDE model's output and

$$L_a = CE(\hat{y}_i^a, y_i), \quad (12)$$

where y_i denotes the ground-truth of anomaly detection. As shown in Eq. (11), we use the cross entropy (CE) loss between $\hat{y}_{i+1}^a$ and $\hat{y}_{i+1}^p$ to distill the knowledge of the anomaly NCDE into the PoA NCDE.

Training with the adjoint method: We train our method using the adjoint sensitivity method [10, 14, 18, 33], which requires a memory of $O(T+H)$ where T is the integral time domain and H is the size of the NCDE's vector field. This method is used in Lines 4 to 5 of Alg. 1. However, our framework uses two NCDEs, which increases the required memory to $O(2T + H_f + H_g)$, where H_f and H_g are the sizes of the vector fields for the two NCDEs, respectively.

To train them, we need to calculate the gradients of each loss w.r.t. the parameters θ_f, θ_g, and θ_c. In this paragraph, we describe how we can space-efficiently calculate them. The gradient to train each parameter can be defined as follows:

$$\begin{aligned}
\nabla_{\theta_f} L_a &= -\int_T^0 \frac{\partial L_a}{\partial \mathbf{h}(t)}^T \frac{\partial f(\mathbf{h}(t),t;\theta_f,\theta_c)}{\partial \theta_f}\frac{dX(t)}{dt}dt,\\
\nabla_{\theta_f} L_{KD} &= -\int_T^0 \frac{\partial L_{KD}}{\partial \mathbf{h}(t)}^T \frac{\partial f(\mathbf{h}(t),t;\theta_f,\theta_c)}{\partial \theta_f}\frac{dX(t)}{dt}dt,\\
\nabla_{\theta_g} L_{KD} &= -\int_T^0 \frac{\partial L_{KD}}{\partial \mathbf{z}(t)}^T \frac{\partial g(\mathbf{z}(t),t;\theta_g,\theta_c)}{\partial \theta_g}\frac{dX(t)}{dt}dt,\\
\nabla_{\theta_c} L_a + L_{KD} &= -\int_T^0 \frac{\partial L_a + L_{KD}}{\partial \mathbf{h}(t)}^T \frac{\partial f(\mathbf{h}(t),t;\theta_f,\theta_c)}{\partial \theta_c}\frac{dX(t)}{dt}dt\\
&\quad -\int_T^0 \frac{\partial L_a + L_{KD}}{\partial \mathbf{z}(t)}^T \frac{\partial g(\mathbf{z}(t),t;\theta_g,\theta_c)}{\partial \theta_c}\frac{dX(t)}{dt}dt.
\end{aligned} \quad (13)$$

3.5 Data Augmentation Methods for Self-supervised Learning

In general, the dataset for anomaly detection do not provide labels for training samples but have labels only for testing samples. Thus,

Reproduce with param s

we rely on the self-supervised learning technique. We mainly use re-sampling methods for data augmentation, and we resort to the following steps for augmenting training samples with anomalous patterns:

(1) Maintain an anomaly ratio $\gamma = \frac{L}{T}$ for each dataset;
(2) The starting points t_{start} of anomaly data points are all designated randomly;
(3) The length vector $\mathbf{l}$ of abnormal data are randomly selected between sequence 100 to sequence 500;
(4) The re-sampled data is randomly selected from the training samples.

L is the total length of abnormal data points, and T is the total length of data points. A detailed data augmentation algorithm is in Appendix.

4 EXPERIMENTS

In this section, we describe our experimental environments and results. All experiments were conducted in the following software and hardware environments: Ubuntu 18.04 LTS, Python 3.8.13, Numpy 1.21.5, Scipy 1.7.3, Matplotlib 3.3.1, PyTorch 1.7.0, CUDA 11.0, NVIDIA Driver 417.22, i9 CPU, and NVIDIA RTX 3090.

4.1 Datasets

Mars Science Laboratory: The mars science laboratory (MSL) dataset is also from NASA, which was collected by a spacecraft en route to Mars. This dataset is a publicly available dataset from NASA-designated data centers. It is one of the most widely used dataset for the anomaly detection research due to the clear distinction between pre and post-anomaly recovery. It is comprised of the health check-up data of the instruments during the journey. This dataset is a multivariate time series dataset, and it has 55 dimensions with an anomaly ratio of approximately 10.72% [20].

Secure Water Treatment: The secure water treatment (SWaT) dataset is a reduced representation of a real industrial water treatment plant that produces filtered water. This data set contains important information about effective measures that can be implemented to avoid or mitigate cyberattacks on water treatment facilities. The data set was collected for a total of 11 days, with the first 7 days collected under normal operating conditions and the subsequent 4 days collected under simulated attack scenarios. SWaT has 51 different values in an observation and an anomaly ratio of approximately 11.98% [15].

Water Distribution: The water distribution (WADI) data set is compiled from the WADI testbed, an extension of the SWaT testbed. It is measured over 16 days, of which 14 days are measured in the normal state and 2 days are collected in the state of the attack scenario. WADI has 123 different values in an observation and an anomaly ratio of approximately 5.99% [29].

4.2 Experimental Settings

4.2.1 Hyperparameters. We list all the detailed hyperparameter setting for baselines and our method in Appendix.

For reproducibility, we report the following best hyperparameters for our method:

(1) In MSL, we train for 300 epochs, a learning rate of $1.0 \times e^{-2}$, a weight decay of $1.0 \times e^{-4}$, and the hidden size of θ_f, θ_g, and θ_c is 256, 512, 256, respectively. Among 256 windows, we detect the window in which abnormal data points exist. The length of each window was set to 30, and the length of the predicted window in precursor anomaly detection was set to 10.
(2) In SWaT, we train for 300 epochs, a learning rate of $1.0 \times e^{-2}$, a weight decay of $1.0 \times e^{-4}$, and the hidden size of θ_f, θ_g, and θ_c is 128, 64, 64, respectively. Among 256 windows, we detect the window in which abnormal data points exist. The length of each window was set to 60, and the length of the predicted window in precursor anomaly detection was set to 20.
(3) In WADI, we train for 300 epochs, a learning rate of $1.0 \times e^{-2}$, a weight decay of $1.0 \times e^{-5}$, and the hidden size of θ_f, θ_g, and θ_c is 128, 128, 256, respectively. Among 256 windows, we detect the window in which abnormal data points exist. The length of each window was set to 100, and the length of the predicted window in precursor anomaly detection was set to 30.

4.2.2 Baselines. We list all the detailed description about baselines in Appendix. We compare our model with the following 16 baselines of 4 categories, including not only traditional methods but also state-of-the-art deep learning-based models as follows:

(1) The classical method category includes OCSVM [43] and Isolation Forest [25].
(2) The clustering-based methods category includes Deep-SVDD [35], ITAD [39], and THOC [38].
(3) The density-estimation-based methods category has LOF [7], DAGMM [54], and MMPCACD [48].
(4) The reconstruction-based methods includes VAR [3], LSTM [20], CL-MPPCA [42], LSTM-VAE [32], BeatGAN [53], OmniAnomaly [40], USAD [4], InterFusion [24], and Anomaly Transformer [46].

4.3 Experimental Results on Anomaly Detection

We introduce our experimental results for the anomaly detection with the following 3 datasets: MSL, SWaT, and WADI. Evaluating the performance with these datasets proves the competence of our model in various fields. We use the Precision, Recall, and F1-score.

4.3.1 Experimental Results on Regular Time Series. Table 1 summarizes the results on the three datasets. The anomaly detection with MSL is one of the most widely used benchmark experiments. Our method, PAD, shows the best F1-score. For this dataset, all classical methods are inferior to other baselines. For SWaT, our experimental results are in Table 1. As summarized, the classical methods are inferior to other baselines. However, unlike in WADI, all baselines except them show similar results. Our method, PAD, shows the best F1-score. For WADI, among the reconstruction-based methods, InterFusion shows the second-best performance. Since this dataset has the smallest anomaly ratio among the three datasets, classical methods and clustering-based methods are not suitable.

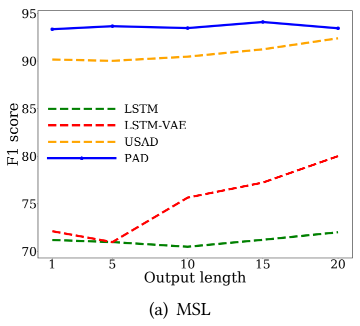
Table 1: Experimental results for the anomaly detection on 3 regular time-series datasets. P, R, and F1 denote Precision, recall and F1 score as %, respectively. The best scores are in bold face.

Datasets		MSL			SWaT			WADI		
Methods		P	R	F1	P	R	F1	P	R	F1
Classical Methods	OCSVM	59.78	86.87	70.82	45.39	49.22	47.23	83.93	46.42	58.64
	Isolation Forest	53.94	86.54	66.45	95.12	58.84	72.71	95.12	58.84	72.71
Clustering-based Methods	Deep-SVDD	91.92	76.63	83.58	80.42	84.45	82.39	83.70	47.88	60.03
	ITAD	69.44	84.09	76.07	63.13	52.08	57.08	92.11	58.79	70.25
	THOC	88.45	90.97	89.69	98.08	79.94	88.09	86.71	92.02	89.29
Density-estimation-based Methods	LOF	81.17	81.44	81.23	72.15	65.43	68.62	87.61	17.92	25.23
	DAGMM	89.60	63.93	74.62	89.92	57.84	70.40	46.95	66.59	55.07
	MMPCACD	81.42	61.31	69.95	82.52	68.29	74.73	88.61	75.84	81.73
Reconstruction-based Methods	VAR	74.68	81.42	77.90	81.59	60.29	69.34	83.97	49.35	61.31
	LSTM	85.45	82.50	83.95	86.15	83.27	84.69	81.43	84.82	83.06
	CL-MPPCA	73.71	88.54	80.44	76.78	81.50	79.07	70.96	75.21	72.86
	LSTM-VAE	85.49	79.94	82.62	76.00	89.50	82.20	98.97	63.77	77.56
	BeatGAN	89.75	85.42	87.53	64.01	87.46	73.92	70.48	72.26	71.34
	OmniAnomay	89.02	86.37	87.67	81.42	84.30	82.83	98.25	64.97	78.22
	USAD	89.36	92.92	91.05	98.51	66.18	79.17	99.47	13.18	23.28
	InterFusion	81.28	92.70	86.62	80.59	85.58	83.01	90.30	92.67	91.02
	Anomaly Transformer	91.82	91.23	91.53	87.32	85.50	86.40	60.86	77.86	68.32
Ours	**PAD** (Anomaly)	94.13	94.71	**92.56**	94.02	93.53	**93.04**	90.84	95.31	**93.02**

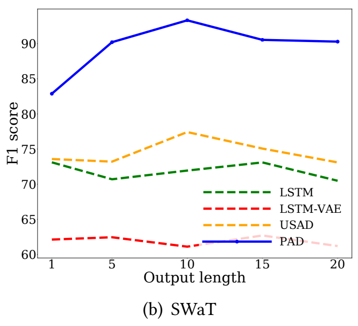
Table 2: Experimental results for the anomaly detection on 3 irregular time-series datasets.

Dropping ratio		30% dropped			50% dropped			70% dropped		
Datasets	Methods	P	R	F1	P	R	F1	P	R	F1
MSL	Isolation Forest	89.81	41.59	52.53	86.97	39.38	50.74	88.03	29.27	38.33
	LOF	78.26	78.71	78.39	74.86	75.39	74.95	73.53	72.59	72.99
	USAD	88.07	84.04	85.92	87.49	68.73	76.26	87.49	74.92	80.33
	Anomaly Transformer	90.49	85.42	87.89	91.26	86.40	88.76	91.75	90.47	91.10
	PAD (Anomaly)	89.09	94.39	**91.66**	94.92	94.63	**92.24**	91.09	94.22	**91.87**
SWaT	Isolation Forest	69.55	49.79	55.06	69.94	35.00	37.16	67.89	24.37	18.83
	LOF	70.85	21.87	12.77	71.20	25.30	20.32	69.34	23.24	16.39
	USAD	87.36	52.64	64.87	84.58	38.44	50.42	87.96	47.19	59.91
	Anomaly Transformer	94.48	77.61	85.22	93.33	84.14	88.50	93.23	82.81	87.71
	PAD (Anomaly)	93.65	93.27	**92.76**	93.89	93.53	**93.07**	92.36	92.47	**92.08**
WADI	Isolation Forest	92.19	48.48	62.33	91.77	26.35	38.03	92.89	12.15	16.23
	LOF	86.42	16.35	21.42	85.78	12.70	15.88	91.12	10.26	13.33
	USAD	84.98	15.70	20.60	83.78	15.95	21.30	75.89	13.19	20.09
	Anomaly Transformer	65.73	100.0	79.32	62.93	84.67	72.20	61.17	77.86	68.52
	PAD (Anomaly)	95.69	95.49	**93.44**	91.21	93.06	**92.10**	90.85	89.41	**90.12**

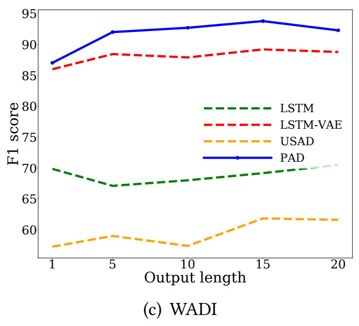
4.3.2 Experimental Results on Irregular Time Series. Table 2 summarizes the results on irregular time series. In order to create challenging irregular environments, we randomly remove 30%, 50% and 70% of the observations in each sequence. Therefore, this is basically an irregular time series anomaly detection. We compare our method, PAD, with the 4 baselines, Isolation Forest, LOF, USAD, and Anomaly Transformer — other baselines are not defined for irregular time series. In addition, it is expected that the presence of more than 30% of missing values in time series causes poor performance for many baselines because it is difficult to understand the highly missing input sequence. In MSL, USAD and Anomaly Transformer shows the reasonable results and also maintains an F1-score around 80% across all the dropping ratio settings. For WADI, all baselines show poor performance when the missing rate is 70%. Surprisingly, our method, PAD, performs not significantly differently from the regular anomaly detection experiments. Our method maintains good performance in the irregular time series setting as well because PAD uses a hidden representation controlled by the continuous path $X(t)$ at every time t. Additionally, our method maintains an F1 score larger than 90% for all the dropping ratios.



(a) MSL



(b) SWaT



(c) WADI

Figure 4: F1 score by varying the output length on 3 datasets.

Table 3: Experimental results for the precursor-of-anomaly detection on 3 regular time-series datasets.

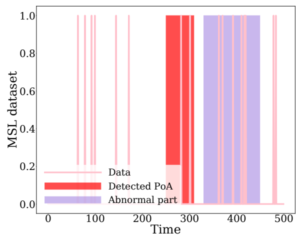
Dataset	MSL			SWaT			WADI		
Methods	P	R	F1	P	R	F1	P	R	F1
LSTM	90.42	59.53	70.49	64.96	80.64	71.94	92.87	55.38	68.02
LSTM-VAE	69.54	83.05	75.65	75.64	55.10	61.11	84.32	91.83	87.91
USAD	90.78	95.27	92.97	84.08	75.07	77.42	94.63	44.10	57.40
PAD (PoA)	91.41	95.61	**93.46**	93.31	93.47	**93.32**	92.71	92.71	**92.71**

4.4 Experimental Results on the precursor-of-Anomaly Detection

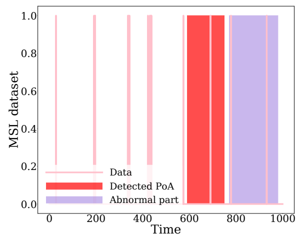
In Table. 3, we introduce our experimental results for the precursor-of-anomaly detection. In order to compare the performance of our method in the precursor-of-anomaly detection task newly proposed in this paper, among the baselines performed in regular time series anomaly detection (cf. Table 1), we selected reconstruction-based methods that allow PoA experimental setting. Therefore, we select the 3 baselines (LSTM, LSTM-VAE, and USAD) that showed good performance in reconstruction-based methods.

4.4.1 Experimental Results on Regular Time Series. As shown in Table 3, USAD shows reasonable performance among the 3 baselines. Especially, in MSL dataset, USAD shows a similar performance to Table 1. Our newly proposed the precursor-of-anomaly task requires predicting patterns or features of future data from input data. Therefore, the reconstruction-based method seems to have shown good performance. However, our method, PAD, shows the best performance in all the 3 datasets. Fig. 5 shows the visualization of experimental results on the anomaly detection and the precursor-of-anomaly detection on all the 3 datasets. In Fig. 5, the part highlighted in purple is the ground truth of the anomalies, the part highlighted in red is the result of PoA detected by PAD. As shown in Fig. 5, our method correctly predicts the precursor-of-anomalies (highlighted in red) before the abnormal parts (highlighted in purple) occur.

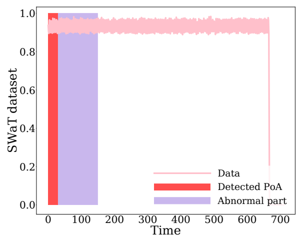
4.4.2 Experimental Results on Irregular Time Series. Table 4 shows the experimental result on the irregular datasets. Among the baselines, USAD has many differences in experimental results depending on the experimental environment. For example, in the WADI dataset, which has a small anomaly ratio(5.99%) among the other datasets, it shows poor performance, and in the MSL data set, USAD



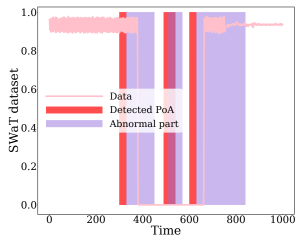
(a) MSL



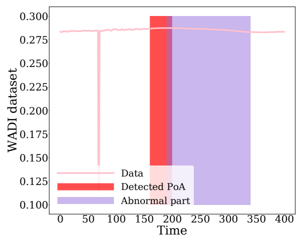
(b) MSL



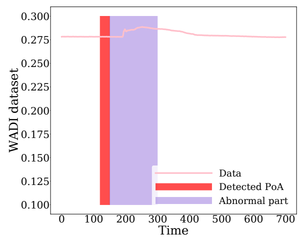
(c) SWaT



(d) SWaT



(e) WADI



(f) WADI

Figure 5: The anomaly detection and the precursor-of-anomaly detection results on 3 datasets.

shows the second-best performance, but the performance deteriorates as the dropping ratio increases. However, our method, PAD, clearly shows the best F1-score for all dropping ratios and all 3 datasets. One outstanding point in our model is that the F1-score is not greatly influenced by the dropping ratio. Consequently, all these results prove that our model shows state-of-the-art performance in both the anomaly and the precursor-of-anomaly detection.

Table 4: Experimental results for the precursor-of-anomaly detection on the 3 irregular time-series datasets.

Dropping ratio		30% dropped			50% dropped			70% dropped		
Datasets	Methods	P	R	F1	P	R	F1	P	R	F1
MSL	LSTM	66.88	81.16	73.29	90.06	50.90	63.28	90.09	48.85	61.41
	LSTM-VAE	66.96	81.75	73.58	90.22	51.29	63.60	90.12	48.78	61.34
	USAD	92.09	88.27	90.03	91.54	78.33	84.04	91.17	69.54	78.27
	PAD (PoA)	91.38	95.04	**93.17**	91.40	95.52	**93.42**	93.50	95.52	**93.84**
SWaT	LSTM	83.57	68.87	72.11	80.15	54.13	58.21	74.56	30.27	27.99
	LSTM-VAE	81.98	56.27	60.22	71.34	35.80	37.61	74.94	30.80	28.80
	USAD	84.12	75.20	77.54	83.58	71.80	74.65	83.15	66.60	70.09
	PAD (PoA)	85.50	77.40	**79.47**	84.15	77.67	**79.53**	90.85	91.07	**90.93**
WADI	LSTM	91.15	46.01	59.95	91.28	25.52	36.68	92.97	18.58	26.29
	LSTM-VAE	89.46	45.66	59.17	91.55	26.91	38.47	93.11	19.27	27.32
	USAD	94.87	39.93	53.05	94.62	34.38	46.95	94.87	28.30	39.49
	PAD (PoA)	92.15	95.49	**93.79**	92.17	96.01	**94.05**	92.15	95.31	**93.70**

Table 5: Ablation study for PAD in a regular setting.

Dataset	MSL			SWaT			WADI		
Metric	P	R	F1	P	R	F1	P	R	F1
Type (i)	94.92	94.63	92.24	94.13	93.67	93.19	90.45	87.15	88.77
Type (ii)	89.74	76.65	82.30	74.94	80.07	72.11	94.57	86.98	89.97
PAD	94.13	94.71	**92.56**	94.02	93.53	**93.04**	90.84	95.31	**93.02**

Table 6: Ablation study for the Precursor-of-Anomaly PAD in a regular setting.

Dataset	MSL			SWaT			WADI		
Metric	P	R	F1	P	R	F1	P	R	F1
Type (i)	91.84	88.93	90.33	90.82	90.67	90.74	91.57	82.47	86.78
Type (ii)	91.37	89.18	90.26	64.96	80.60	71.94	93.12	89.06	90.95
PAD (PoA)	91.41	95.61	**93.46**	93.31	93.47	**93.32**	92.71	92.71	**92.71**

5 ABLATION AND SENSITIVITY STUDIES

5.1 Ablation Study on Multi-task Learning

To prove the efficacy of our multi-task learning on the anomaly and the precursor-of-anomaly detection, we conduct ablation studies. There are 2 tasks in our multi-task learning: the anomaly detection, and the precursor-of-anomaly detection tasks. We remove one task to build an ablation model. For the ablation study on anomaly detection, there are 2 ablation models in terms of the multi-task learning setting: i) without the precursor-of-anomaly detection, and ii) with anomaly detection only. For the ablation study on the precursor-of-anomaly detection, 2 ablation models are defined in the exactly same way. Table 5 and Table 6 show the results of the ablation studies in the regular time series setting for 'PAD (anomaly)' and 'PAD (PoA),' respectively. When we remove task(anomaly detection or PoA detection) from the multi-task learning, there is notable degradation in performance. Therefore, our multi-task learning design is required for good performance in both the anomaly and the precursor-of-anomaly detection.

5.2 Sensitivity to Output Sequence Length

We also compare our method with USAD, OmniAnomaly, and THOC by varying the length of output of the precursor-of-anomaly detection, during the multi-task learning process. After fixing the input length of MSL, SWaT, and WADI to 30, we vary the output length in $\{1, 5, 10, 15, 20\}$. As shown in Fig. 4, our proposed method consistently outperforms others. As the output length increases, it becomes more difficult to predict in general, but PAD shows excellent performance regardless of the output length. In the MSL and WADI datasets, most baselines show similar performances regardless of the output length. However, in SWaT, there is a performance difference according to the output length, and this phenomenon appears similarly for the baselines.

6 CONCLUSION

Recently, many studies have been conducted on time series anomaly detection. However, most of the methods have been conducted only for existing anomaly detection methods. In this paper, we first propose a task called the precursor-of-anomaly (PoA) detection. We define PoA detection as the task of predicting future anomaly detection. This study is necessary in that many risks can be minimized by detecting risks in advance in the real world. In addition, we combine multi-task-learning and NCDE architecture to perform both anomaly detection, and PoA detection and achieve the best performance through task-specific-parameter sharing. Additionally, we propose a novel dual co-evolving NCDE structure. Two NCDEs perform anomaly detection and PoA detection tasks. Our experiments with the 3 real-world datasets and 17 baseline methods successfully prove the efficacy of the proposed concept. In addition, our visualization of anomaly detection results delivers how our proposed method works. In the future work, we plan to conduct unsupervised precursor-of-anomaly detection research since the time series data augmentation method requires a pre-processing step.

ACKNOWLEDGEMENTS

Noseong Park is the corresponding author. This work was supported by the Yonsei University Research Fund of 2021, and the

Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. 2020-0-01361, Artificial Intelligence Graduate School Program (Yonsei University), and (No.2022-0-00857, Development of AI/data-based financial/economic digital twin platform,10%) and (No.2022-0-00113, Developing a Sustainable Collaborative Multi-modal Lifelong Learning Framework, 45%),(2022-0-01032, Development of Collective Collaboration Intelligence Framework for Internet of Autonomous Things, 45%).

REFERENCES

[1] Shikha Agrawal and Jitendra Agrawal. 2015. Survey on anomaly detection using data mining techniques. *Procedia Computer Science* 60 (2015), 708–713.

[2] Jinwon An and Sungzoon Cho. 2015. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE* 2, 1 (2015), 1–18.

[3] Oliver D. Anderson and M. G. Kendall. 1976. Time-Series. 2nd edn. *The Statistician* 25 (1976), 308.

[4] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A Zuluaga. 2020. Usad: Unsupervised anomaly detection on multivariate time series. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3395–3404.

[5] Emel Ay, Maxime Devanne, Jonathan Weber, and Germain Forestier. 2022. A study of knowledge distillation in fully convolutional network for time series classification. In *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[6] Anshuman Bhardwaj, Shaktiman Singh, Lydia Sam, Pawan K Joshi, Akanksha Bhardwaj, F Javier Martín-Torres, and Rajesh Kumar. 2017. A review on remotely sensed land surface temperature anomaly as an earthquake precursor. *International journal of applied earth observation and geoinformation* 63 (2017), 158–166.

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

[8] Chris U. Carmona, François-Xavier Aubet, Valentin Flunkert, and Jan Gasthaus. 2022. Neural Contextual Anomaly Detection for Time Series. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, Lud De Raedt (Ed.). International Joint Conferences on Artificial Intelligence Organization, 2843–2851. https://doi.org/10.24963/ijcai.2022/394 Main Track.

[9] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2010. Anomaly detection for discrete sequences: A survey. *IEEE transactions on knowledge and data engineering* 24, 5 (2010), 823–839.

[10] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural Ordinary Differential Equations. In *NeurIPS*.

[11] Shijie Chen, Yu Zhang, and Qiang Yang. 2021. Multi-task learning in natural language processing: An overview. *arXiv preprint arXiv:2109.09138* (2021).

[12] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Sal Stolfo. 2002. A geometric framework for unsupervised anomaly detection. In *Applications of data mining in computer security*. Springer, 77–101.

[13] Dipak Ghosh, Argha Deb, and Rosalima Sengupta. 2009. Anomalous radon emission as precursor of earthquake. *Journal of Applied Geophysics* 69, 2 (2009), 67–81.

[14] Mike Giles and Niles Pierce. 2000. An Introduction to the Adjoint Approach to Design. *Flow, Turbulence and Combustion* 65 (2000), 393–415. https://doi.org/10.1023/A:1011430410075

[15] Jonathan Goh, Sridhar Adepu, Khurum Nazir Junejo, and Aditya Mathur. 2017. A dataset to support research in the design of secure water treatment systems. In *Critical Information Infrastructures Security: 11th International Conference, CRITIS 2016, Paris, France, October 10–12, 2016, Revised Selected Papers 11*. Springer, 88–99.

[16] Markus Goldstein and Seiichi Uchida. 2016. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one* 11, 4 (2016), e0152173.

[17] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. https://doi.org/10.48550/ARXIV.1406.2661

[18] William Hager. 2000. Runge-Kutta Methods in Optimal Control and the Transformed Adjoint System. *Numer. Math.* 87 (2000), 247–282. https://doi.org/10.1007/s002110000178

[19] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[20] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. 2018. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 387–395.

[21] Patrick Kidger, James Morrill, James Foster, and Terry Lyons. 2020. Neural Controlled Differential Equations for Irregular Time Series. In *NeurIPS*.

[22] Florian Knorn and Douglas J Leith. 2008. Adaptive kalman filtering for anomaly detection in software appliances. In *IEEE INFOCOM Workshops 2008*. IEEE, 1–6.

[23] Chun-Liang Li, Kihyuk Sohn, Jinsung Yoon, and Tomas Pfister. 2021. CutPaste: Self-Supervised Learning for Anomaly Detection and Localization. https://doi.org/10.48550/ARXIV.2104.04015

[24] Zhihan Li, Youjian Zhao, Jiaqi Han, Ya Su, Rui Jiao, Xidao Wen, and Dan Pei. 2021. Multivariate time series anomaly detection and interpretation using hierarchical inter-metric and temporal embedding. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3220–3230.

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

[26] Shikun Liu, Edward Johns, and Andrew J Davison. 2019. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 1871–1880.

[27] Yi Liu, Sahil Garg, Jiangtian Nie, Yang Zhang, Zehui Xiong, Jiawen Kang, and M Shamim Hossain. 2020. Deep anomaly detection for time-series data in industrial IoT: A communication-efficient on-device federated learning approach. *IEEE Internet of Things Journal* 8, 8 (2020), 6348–6358.

[28] Terry Lyons, M. Caruana, and T. Lévy. 2004. *Differential Equations Driven by Rough Paths*. Springer. École D'Eté de Probabilités de Saint-Flour XXXIV - 2004.

[29] Aditya P Mathur and Nils Ole Tippenhauer. 2016. SWaT: A water treatment testbed for research and training on ICS security. In *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*. IEEE, 31–36.

[30] Sky McKinley and Megan Levine. 1998. Cubic spline interpolation. *College of the Redwoods* 45, 1 (1998), 1049–1060.

[31] Ramin Moghaddass and Shuangwen Sheng. 2019. An anomaly detection framework for dynamic systems using a Bayesian hierarchical framework. *Applied energy* 240 (2019), 561–582.

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

[33] L.S. Pontryagin, E.F. Mishchenko, V.G. Boltyanski, and R.V. Gamkrelidze. 1962. *The mathematical theory of optimal processes*. Interscience Publishers.

[34] Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098* (2017).

[35] Lukas Ruff, Robert A Vandermeulen, Nico Görnitz, Alexander Binder, Emmanuel Müller, Klaus-Robert Müller, and Marius Kloft. 2019. Deep semi-supervised anomaly detection. *arXiv preprint arXiv:1906.02694* (2019).

[36] Mohammadreza Salehi, Ainaz Eftekhar, Niousha Sadjadi, Mohammad Hossein Rohban, and Hamid R. Rabiee. 2020. Puzzle-AE: Novelty Detection in Images through Solving Puzzles. https://doi.org/10.48550/ARXIV.2008.12959

[37] Arun K Saraf, Vineeta Rawat, Swapnamita Choudhury, Sudipta Dasgupta, and Josodhir Das. 2009. Advances in understanding of the mechanism for generation of earthquake thermal precursors detected by satellites. *International Journal of Applied Earth Observation and Geoinformation* 11, 6 (2009), 373–379.

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

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

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

[41] Jihoon Tack, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin. 2020. CSI: Novelty Detection via Contrastive Learning on Distributionally Shifted Instances. https://doi.org/10.48550/ARXIV.2007.08176

[42] Shahroz Tariq, Sangyup Lee, Youjin Shin, Myeong Shin Lee, Okchul Jung, Daewon Chung, and Simon S Woo. 2019. Detecting anomalies in space using multivariate convolutional LSTM with mixtures of probabilistic PCA. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2123–2133.

[43] David MJ Tax and Robert PW Duin. 2004. Support vector data description. *Machine learning* 54, 1 (2004), 45–66.

[44] Chee-Wooi Ten, Junho Hong, and Chen-Ching Liu. 2011. Anomaly detection for cybersecurity of the substations. *IEEE Transactions on Smart Grid* 2, 4 (2011), 865–873.

[45] Di Wu, Zhongkai Jiang, Xiaofeng Xie, Xuetao Wei, Weiren Yu, and Renfa Li. 2019. LSTM learning with Bayesian and Gaussian processing for anomaly detection in industrial IoT. *IEEE Transactions on Industrial Informatics* 16, 8 (2019), 5244–5253.

[46] Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. 2021. Anomaly transformer: Time series anomaly detection with association discrepancy. *arXiv*

preprint arXiv:2110.02642 (2021).
[47] Qing Xu, Zhenghua Chen, Mohamed Ragab, Chao Wang, Min Wu, and Xiaoli Li. 2022. Contrastive adversarial knowledge distillation for deep model compression in time-series regression tasks. *Neurocomputing* 485 (2022), 242–251.
[48] Takehisa Yairi, Naoya Takeishi, Tetsuo Oda, Yuta Nakajima, Naoki Nishimura, and Noboru Takata. 2017. A data-driven health monitoring method for satellite housekeeping data based on probabilistic clustering and dimensionality reduction. *IEEE Trans. Aerospace Electron. Systems* 53, 3 (2017), 1384–1401.
[49] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*. 6023–6032.
[50] Houssam Zenati, Chuan Sheng Foo, Bruno Lecouat, Gaurav Manek, and Vijay Ramaseshan Chandrasekhar. 2018. Efficient gan-based anomaly detection. *arXiv preprint arXiv:1802.06222* (2018).
[51] Meng Zhang, Anand Raghunathan, and Niraj K Jha. 2013. MedMon: Securing medical devices through wireless monitoring and anomaly detection. *IEEE Transactions on Biomedical circuits and Systems* 7, 6 (2013), 871–881.
[52] Yu Zhang and Qiang Yang. 2021. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering* 34, 12 (2021), 5586–5609.
[53] Bin Zhou, Shenghua Liu, Bryan Hooi, Xueqi Cheng, and Jing Ye. 2019. BeatGAN: Anomalous Rhythm Detection using Adversarially Generated Time Series.. In *IJCAI*. 4433–4439.
[54] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. 2018. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International conference on learning representations*.

A BASELINES

We list all the hyperparameter settings for baselines and our method in Appendix. We compare our model with the following 16 baselines of 5 categories, including not only traditional methods but also state-of-the-art deep learning-based models as follows:

(1) The classical method category includes the following algorithms:
  (a) OCSVM [43] aims to find optimized support vectors that can accurately explain given data.
  (b) Isolation Forest [25] is basically from a decision tree. It detects anomalies based on density.
(2) The clustering-based method category has the following methods:
  (a) Deep support vector data description (Deep-SVDD) [35] is a model that applies deep learning to SVDD for the anomaly detection. This method detects anomalies that are far from the compressed representation of normal data.
  (b) Integrative tensor-based anomaly detection (ITAD) [39] is a tensor-based model. This model uses not only tensor decomposition but also k-means clustering to distinguish normal and abnormal.
  (c) Temporal hierarchical one-class (THOC) [38] utilizes a dilated RNN with skip connections to capture dynamics of time series in multiple scales. Multi-resolution temporal clusters are helpful for the anomaly detection.
(3) The density-estimation-based methods are as follows:
  (a) LOF [7] considers the relative density of the data and considers data with low density as abnormal data points.
  (b) Deep auto-encoding gaussian mixture model (DAGMM) [54] consists of a compression network and an estimation network. Each network measures information necessary for the anomaly detection and the likelihood of the information.
  (c) MPPCACD [48] is a kind of GMM and employs probabilistic dimensionality reduction and clustering to detect anomalies.
(4) The reconstruction-based method category has the following methods:
  (a) VAR [3] adapted ARIMA to anomaly detection. Detect anomalies as prediction errors for future observations.
  (b) LSTM [20] is a kind of RNN and is widely used for time series forecasting, because it is an algorithm for learning long-time dependencies.
  (c) CL-MPPCA [42] exploits convolutional LSTM to forecast future observations and mixtures of probabilistic principal component analyzers (MPPCA) to complement the convolutional LSTM.
  (d) LSTM-VAE [32] is a model composed of an LSTM-based variational autoencoder(VAE). The decoder reconstructs the expected distribution of input. An anomaly score is measured with the estimated distribution.
  (e) BeatGAN [53] trains an autoencoder with a discriminator and an adversarial loss like GANs [17]. Reconstruction errors are used as anomaly scores.
  (f) OmniAnomaly [40] model the temporal dependency between stochastic variables by combining GRU, VAE, and planar normalizing flows. Detect anomalies based on the reconstruction probability.
  (g) Unsupervised anomaly detection (USAD) [4] is an autoencoder architecture based on adversarial training. This method detects anomalies based on reconstruction errors.
  (h) InterFusion [24] uses hierarchical VAE with two stochastic latent variables for inter-metric or temporal embeddings. Furthermore, MCMC-based method helps to obtain more reasonable reconstructions.
  (i) Anomaly Transformer [46] incorporates the innovative "Anomaly-Attention" mechanism to enhance the distinguishability, which computes the association discrepancy, and employs a minimax strategy.

Table 7: The best architecture of the CDE function f for MSL. FC, ϕ,ρ stands for the fully-connected layer, the rectified linear unit (ReLU), and the hyperbolic tangent (tanh), respectively.

Design	Layer	Input	Output
ϕ(FC)	1	256×64	256×256
ϕ(FC)	2	256×256	256×256
ϕ(FC)	3	256×256	256×256
ϕ(FC)	4	256×256	256×256
FC	5	256×256	$256 \times 3,520$

Table 8: The best architecture of the CDE function g for MSL

Design	Layer	Input	Output
ϕ(FC)	1	256×64	256×512
ϕ(FC)	2	256×512	256×512
ϕ(FC)	3	256×512	256×512
ϕ(FC)	4	256×512	256×512
FC	5	256×512	$256 \times 3,520$

Table 9: The best architecture of the shared CDE function c for MSL

Design	Layer	Input	Output
ϕ(FC)	1	256×64	256×64
ρ(FC)	2	256×256	$256 \times 3,520$

B HYPERPARAMETERS FOR OUR MODEL

We test the following common hyperparameters for our method:

(1) In MSL, we train for 100 epochs, a learning rate of $\{1.0 \times e^{-2}, 1.0 \times e^{-3}, 1.0 \times e^{-4}\}$, a weight decay of $\{1.0 \times e^{-3}, 1.0 \times e^{-4}, 1.0 \times e^{-5}\}$, and a size of hidden vector size is $\{19, 29, 39, 49\}$.
(2) In SWaT, we train for 100 epochs, a learning rate of $\{1.0 \times e^{-2}, 1.0 \times e^{-3}, 1.0 \times e^{-4}\}$, a weight decay of $\{1.0 \times e^{-3}, 1 \times e^{-4}, 1 \times e^{-5}\}$, and a size of hidden vector size is $\{39, 49, 59\}$.
(3) In WADI, we train for 100 epochs, a learning rate of $\{1.0 \times e^{-2}, 1.0 \times e^{-3}, 1.0 \times e^{-4}\}$, a weight decay of $\{1.0 \times e^{-3}, 1.0 \times e^{-4}, 1.0 \times e^{-5}\}$, and a size of hidden vector size is $\{29, 39, 49, 59\}$.

In Table 7 to 15, we clarify the network architecture of the CDE functions, f,g, and c.

Table 10: The best architecture of the CDE function f for SWaT.

Design	Layer	Input	Output
ϕ(FC)	2	256×64	256×128
ϕ(FC)	3	256×128	256×128
ϕ(FC)	4	256×128	256×128
ϕ(FC)	5	256×128	256×128
FC	6	256×128	$256 \times 3,264$

Table 11: The best architecture of the CDE function g for SWaT.

Design	Layer	Input	Output
ϕ(FC)	2	256×64	256×64
ϕ(FC)	3	256×64	256×64
ϕ(FC)	4	256×64	256×64
ϕ(FC)	5	256×64	256×64
FC	6	256×64	$256 \times 3,264$

Table 12: The best architecture of the shared CDE function c for SWaT.

Design	Layer	Input	Output
ϕ(FC)	1	256×64	256×64
ρ(FC)	2	256×64	$256 \times 3,264$

Table 13: The best architecture of the CDE function f for WADI.

Design	Layer	Input	Output
ϕ(FC)	1	256×16	256×128
ϕ(FC)	2	256×128	256×128
ϕ(FC)	3	256×128	256×128
ϕ(FC)	4	256×128	256×128
FC	5	256×128	$256 \times 1,968$

C HYPERPARAMETERS FOR BASELINES

For the best outcome of baselines, we conduct hyperparmeter search for them based on the recommended hyperparameter set from each papers.

(1) Classical methods: For OCSVM, we use the RBF kernel. For Isolation Forest, we use the base estimator of 100 in the ensemble, and choose the number of trees in $\{25, 100\}$.
(2) Clustering-based methods: For Deep SVDD and ITAD, we use a learning rate of $\{1.0 \times e^{-3}, 1.0 \times e^{-4}\}$ and a hidden vector dimension of $\{32, 64, 128\}$ and For THOC, we use a number of hidden units in $\{32, 64, 84\}$, and we use its default hyperparameters.
(3) Density-estimation methods: For LOF, we use the number of neighbors in $\{1, 3, 5, 7, 12\}$. For DAGMM and MMPCACD, we follow those default hyperparameters.

Table 14: The best architecture of the CDE function g for WADI.

Design	Layer	Input	Output
ϕ(FC)	1	256×16	256×128
ϕ(FC)	2	256×128	256×128
ϕ(FC)	3	256×128	256×128
ϕ(FC)	4	256×128	256×128
FC	5	256×128	$256 \times 1,968$

Table 15: The best architecture of the shared CDE function c for WADI.

Design	Layer	Input	Output
ϕ(FC)	1	256×16	256×256
ρ(FC)	2	256×256	$256 \times 1,968$

(4) Reconstruction-based methods: For LSTM, we use a learning rate of $\{1.0 \times e^{-3}, 1.0 \times e^{-4}\}$ and a hidden vector dimension of $\{80, 100, 120\}$. For LSTM-VAE, BeatGAN, and OmniAnomaly, we use a learning rate of $\{1.0 \times e^{-4}, 1.0 \times e^{-5}\}$ and a hidden vector dimension of $\{32, 64, 128\}$. For USAD, we use a learning rate of $\{1.0 \times e^{-3}, 1.0 \times e^{-4}\}$ and a hidden vector dimension of $\{32, 64, 128\}$, and follow other default hyperparameters in USAD. For InterFusion, a hidden vector dimension of $\{128, 256, 512\}$, and follow other default hyperparameters in InterFusion. For Anomaly Transformer, we follow all hyperparameters in Anomaly Transformer.

D DATA PREPROCESSING DETAILS

We use the following datasets to compare PAD with other methods:

(1) Mars Science Laboratory: MSL is a public dataset from NASA. MSL is licensed under the following license: https://github.com/khundman/telemanom/blob/master/LICENSE.txt
(2) Secure Water Treatment: SWaT is licensed under the following license: https://itrust.sutd.edu.sg/testbeds/secure-water-treatment-swat/
(3) Water Distribution: WADI is a public dataset from NASA. WADI is licensed under the following license: https://itrust.sutd.edu.sg/testbeds/water-distribution-wadi/

Since our method resorts to a self-supervised multi-task learning approach, we augmented training samples with abnormal patterns and its detailed process is in Alg. ??. The augmentation method is similar to other popular augmentation methods for images, e.g., CutMix [49]. There is a long training sequence $\mathbf{x}_{0:T} = \{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_T\}$. We apply the augmentation method to the raw sequence $\mathbf{x}_{0:T}$ before segmenting it into windows. We randomly copy existing l observations from a random location r to a target location s. In general, the ground-truth anomaly pattern is unknown in each dataset. Although our copy-and-paste augmentation method is simple, our experimental results prove its effectiveness. At the same time, we also believe that there will be better augmentation methods.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009