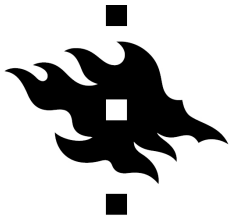

UNIVERSITY OF HELSINKI

A Study of Anomaly Detection Algorithms in Non-Stationary Time Series

Master’s Programme in Computer Science, Software Track

Master's thesis

Author:

Nanqing Jing

Supervisor(s):

Professor Jiaheng Lu

05.05.2025

Helsinki

Faculty: Faculty of Science
Degree programme: Master's Programme in Computer Science
Study track: Software
Author: Nanqing Jing
Title: A Study of Anomaly Detection Algorithms in Non-Stationary Time Series
Level: Master's thesis
Month and year: May 2025
Number of pages: 51
Keywords: Anomaly Detection, Time Series, Cumulative Sum Control Chart (CUSUM), Copula-Based Outlier Detection (COPOD), Isolation Forest (iForest), One-Dimensional Convolutional Neural Network (1D-CNN)
Supervisor or supervisors: Jiaheng Lu
Where deposited: Helsinki University Library
Additional information:
Abstract:
In recent years, anomaly detection (AD) has attracted wide attention in areas such as financial risk control, medical diagnosis, and industrial process monitoring, especially for time series data. Compared to stationary time series, non-stationary time series often exhibit changes in statistical properties like mean and variance over time, which makes traditional statistical modeling methods difficult to apply directly. In addition, anomalies are typically rare and hard to label, further increasing the difficulty of anomaly detection tasks. Therefore, building robust and effective anomaly detection models in complex and dynamic data environments has become a major research focus.

This study focuses on unsupervised anomaly detection in non-stationary time series and uses the ECG5000 dataset as a unified platform. It systematically analyses four mainstream detection methods: statistical control-based CUSUM, density estimation-based COPOD, random split-based iForest, and the deep learning-based 1D-CNN. Based on a comparative analysis of their modeling principles and performance differences, I further propose three improvements: (1) EWSW-CUSUM, which incorporates a sliding window and exponential weighting to enhance sensitivity to local changes and significantly reduce false positives; (2) SWA-COPOD, which captures contextual relationships among anomalies through multi-sample window modeling and prediction aggregation, addressing the original COPOD's limited sensitivity to structural anomalies; and (3) PCA-iForest, which applies dimensionality reduction to decrease invalid splits caused by redundancy in high-dimensional data. After hyperparameter tuning on the validation set and comprehensive evaluation on the test set, results show that both EWSW-CUSUM and SWA-COPOD achieve F1-scores comparable to or better than 1D-CNN, with advantages in both efficiency and interpretability.

This work integrates theoretical analysis and empirical validation, emphasizing the role of sliding windows, contextual modeling, and principal component analysis in anomaly detection for non-stationary sequences. The proposed algorithms remain robust under sample scarcity and unknown data distributions. Future work may extend these optimization mechanisms to other algorithms or apply the proposed algorithms to multivariate time series and other datasets, further investigating the potential of ensemble learning and semi-supervised approaches in this field.

Table of contents

1 Introduction

1.1 Background and Motivation

In the era of big data and artificial intelligence, data has become increasingly important. It serves both as input and output and exists in the form of data streams across a wide range of scenarios. Among various types of data, time series data refers to a sequence of observations collected either continuously or discretely over time, and it can be represented as:

$$S = \{(X_i, T_i)\}, i = 1, 2, \ldots \quad (1)$$

Each observation in the sequence can be represented by (X_i, T_i), where $X_i = (X_{i1}, X_{i2}, \ldots, X_{id}) \in \mathbb{R}^d$ is a d-dimensional vector observed at time T_i.

Time series data have characteristics such as temporal dependency, high dimensionality, and non-stationarity [1-3]. These properties make it fundamentally different from other types of data. In practice, time series data is widely used in various domains, such as finance (e.g., stock prices and trading volumes), healthcare (e.g., blood pressure, glucose levels, ECG), the Internet of Things (e.g., sensor data from smart devices), economics (e.g., GDP, unemployment rates), and natural sciences (e.g., temperature, precipitation) [4-8].

Due to the importance and complexity of time series data, a wide range of analytical approaches have emerged, among which anomaly detection (AD) has received significant attention. Over the years, many definitions of AD tasks have been proposed. According to Chandola et al. (2009), anomaly detection essentially involves identifying samples in a dataset that significantly deviate from the majority of the data distribution [9].

Anomalies can arise for various reasons depending on the context, such as mechanical failures, system errors, changes in behaviour patterns, fraudulent activities, malicious attacks, measurement errors, and noise. These factors make anomaly detection highly meaningful. For example, in the financial domain, AD can help detect credit card fraud; in cybersecurity, it can be used to identify abnormal patterns; and in industrial manufacturing, it can detect product defects in time to improve product quality and production efficiency [9-11].

In the real world, non-stationary time series—whose statistical properties like mean and variance change over time—are more common. Anomalies in such data are often rare and difficult to label. Therefore, unsupervised methods are generally a more suitable choice for anomaly detection in these scenarios.

Based on the above factors, I chose the task of unsupervised anomaly detection in non-stationary time series as the research topic of my master's thesis.

1.2 Related Work

Over the past few decades, with the development of big data, the Internet of Things, and artificial intelligence, both academia and industry have conducted extensive research on anomaly detection. Relevant theories have already been applied to industrial processes, signal processing, cybersecurity, and other fields.

Even within time series data, the characteristics of the data can vary greatly, such as data types, dimensionality, and the presence or absence of labels. In addition, the goals of anomaly detection differ: some scenarios rely on offline data for decision support, while others require real-time analysis of streaming data. To address the challenges posed by diverse data characteristics and application scenarios, a wide range of techniques and algorithms have been proposed for solving AD problems. These methods can generally be categorized into two groups: statistical methods and machine learning methods.

Statistical methods detect anomalies in time series based on probability distributions, statistical tests, or other statistical measures. These methods typically rely on strong assumptions about data distributions but are generally efficient in computation.

In 1954, E.S. Page proposed the Cumulative Sum Control Chart (CUSUM) technique, which was the first to use cumulative sums to detect small shifts in processes [12]. Essentially, CUSUM is a change point detection (CPD) method. However, CPD and AD tasks in time series are closely related in both techniques and application scenarios. As a result, CUSUM has inspired many improvements and variants for solving AD tasks [15, 16].

The Gaussian Mixture Model (GMM) is a type of density-based statistical model that has been widely used in time series AD tasks, particularly for detecting anomalies in

complex, multimodal data. Jiahui Qu et al. (2012) proposed a GMM-based method called GMMD, which achieved excellent performance on hyperspectral data [17].

Both standard CUSUM and GMM are parametric methods, meaning they depend on predefined probability distributions. In real-world scenarios, however, data distributions are often difficult to estimate or may change over time, limiting the applicability of such methods. To address this, Goldstein et al. (2012) proposed a non-parametric anomaly detection algorithm called HBOS, which offers extremely fast computation [18].

Machine learning methods detect changes by training models to learn the characteristics of time series data, often using classification, clustering, and deep learning techniques. These approaches are suitable for scenarios with uncertain data distributions, although they generally have higher computational complexity compared to statistical methods.

Schölkopf et al. (2001) proposed the concept of Support Vector Data Description (SVDD), a method based on Support Vector Machines (SVM), which estimates the primary distribution region of high-dimensional data and identifies anomalies that deviate from it [19]. SVDD has become a classic algorithm in anomaly detection and is widely applied in industrial processes, healthcare, and finance [20-22].

With the increasing availability and performance of computational resources, deep learning techniques have also been widely adopted for solving AD problems. LSTM-based autoencoder models (LSTM-AE) have been extensively studied and improved. Mahmoud Said Elsayed et al. (2020) employed OC-SVM and LSTM-AE to build a hybrid model for anomaly detection in network environments [23]. Rakesh Shrestha et al. (2024) integrated LSTM-AE with federated learning to detect anomalies in smart grids [24]. Originally designed for two-dimensional image processing, Convolutional Neural Networks (CNNs) are now increasingly used for anomaly detection in time series data through one-dimensional CNN (1D-CNN) techniques [25, 41-42].

1.3 Research Objectives and Contributions

Although the "Related Work" section has introduced various anomaly detection methods, their applicability is often limited by dataset characteristics or algorithm design. The goal of this study is to systematically analyse the performance of several types of unsupervised AD methods on a unified dataset, and to explore improvements to existing algorithms so that they can achieve robust detection performance even under unfavourable data conditions. These exploratory enhancements may offer directions for future research and provide insights for model optimization in practical applications.

The contributions of this study are as follows:

- Selected the ECG5000 dataset as the time series data for training and testing, which posed two major challenges for anomaly detection (AD) tasks:

  (1) Non-stationarity of the data, making it unsuitable for statistical methods relying on stable distributions

  (2) An imbalanced train-test ratio (1:9), which tests the generalization ability of the methods under limited training data.

- Systematically compared four representative unsupervised AD algorithms—CUSUM, COPOD, iForest, and 1D-CNN—from both theoretical and practical perspectives.

- Proposed enhancements to the three baseline algorithms:

  (1) EWSW-CUSUM for robust anomaly detection without requiring strict distributional assumptions.

  (2) SWA-COPOD to compensate for COPOD's lack of contextual awareness in detecting anomalies.

  (3) PCA optimization applied to iForest.

- Conducted a comprehensive performance evaluation of both standard and proposed methods, demonstrating the improved robustness and generalizability of the proposed algorithms.

The overall research structure is shown in Figure 1.

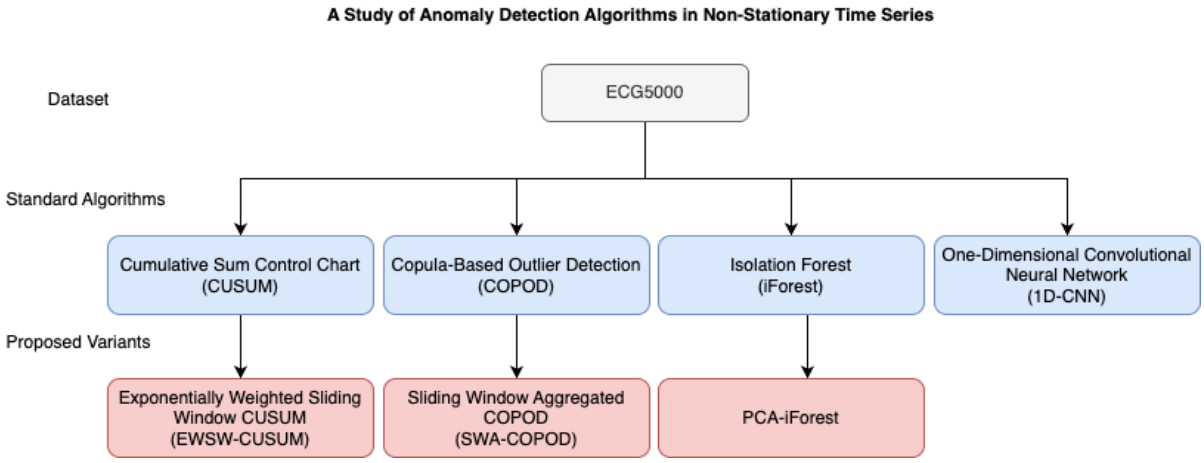

Figure 1. Research structure

2 Dataset

2.1 ECG5000 Dataset

In this study, I used the ECG5000 dataset, which is derived from the BIDMC Congestive Heart Failure Database. This source contains approximately 20 hours of electrocardiogram (ECG) recordings. The original signals were preprocessed by Y. Chen and E. Keogh, and 5000 heartbeats were randomly selected to form the ECG5000 dataset [26, 27]. Each sequence contains 140 time points. The dataset consists of two parts: a training set with 500 samples and a test set with 4500 samples. Each sequence was standardized using z-normalization based on its own mean and standard deviation. Every heartbeat is assigned to one of five subclasses, as shown in Table 1.

Table 1: All categories in the ECG5000 dataset

Class	Subclass & Label	Meaning
Normal	N (1)	Normal
Abnormal	R-on-T (2)	R-on-T PVC
	R-on-T PVC (3)	Polymorphic ventricular tachycardia
	SP (4)	Supraventricular premature beats
	UB (5)	Unclassified beats

Table 2 lists the number of heartbeats in each category (Class 1–5) in the original training and test sets.

Table 2: Number of heartbeats per class in the ECG5000 training and test sets

Class	Subclass & Label	Training Set	Test Set	Total
Normal	N (1)	292	2627	2919
Abnormal	R-on-T (2)	177	1590	2081
	R-on-T PVC (3)	10	86	
	SP (4)	19	175	
	UB (5)	2	22	
Total		500	4500	5000

2.2 Exploratory Data Analysis

To gain a better understanding of the dataset and prepare for model building, it is necessary to perform exploratory analysis. As shown in Figure 2, the majority of heartbeats in ECG5000 (58.37%) correspond to normal cycles. Among the abnormal samples, most ($\frac{35.35\%}{1-58.37\%} = 84.91\%$) are associated with the R-on-T phenomenon (Class 2).

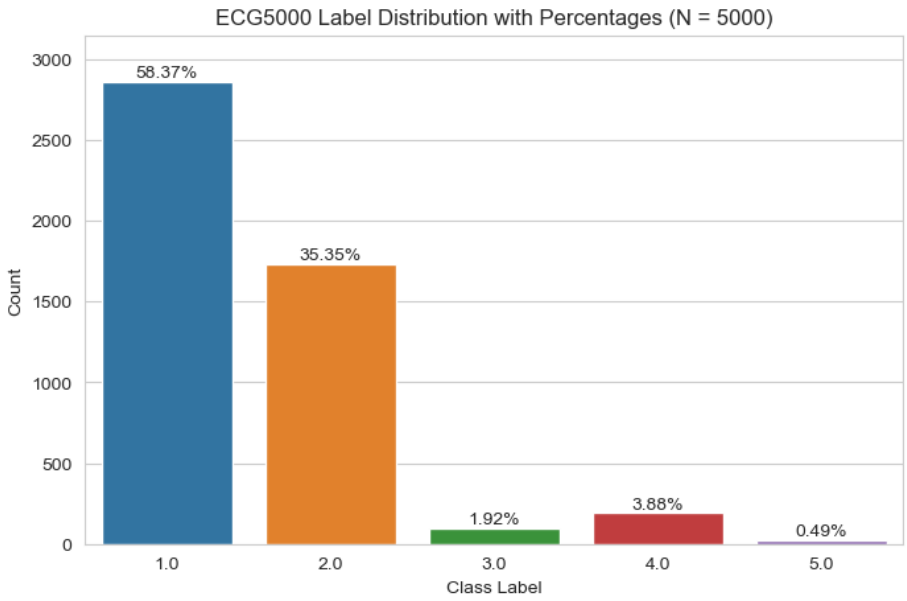

Figure 2. Distribution of different classes in the dataset (Class 1 = normal, Classes 2–5 = abnormal)

To visually highlight the differences between categories, I plotted the mean curves and standard deviation bands for each class, as shown in Figure 3. Class 1 (normal) clearly exhibits the smallest standard deviation, and its trough between time points 0–20 is the deepest among all classes (approaching -4). Classes 2 and 3 show abnormal troughs in the final 10 time points, but Class 2 features a sharp upward spike. Class 4 resembles Class 3, though its terminal trough is shallower and accompanied by a noticeably wider standard deviation band. Class 5 consistently has the widest standard deviation band throughout the entire sequence, indicating high variability. The final subplot overlays all class curves, revealing significant differences in both mean and standard deviation between the normal class (Class 1) and the abnormal classes (Class 2-5) in the intervals [0, 20] and [100, 140]. This preliminary analysis suggests that the beginning and end segments of the heartbeat time series may be potential candidates for anomalies.

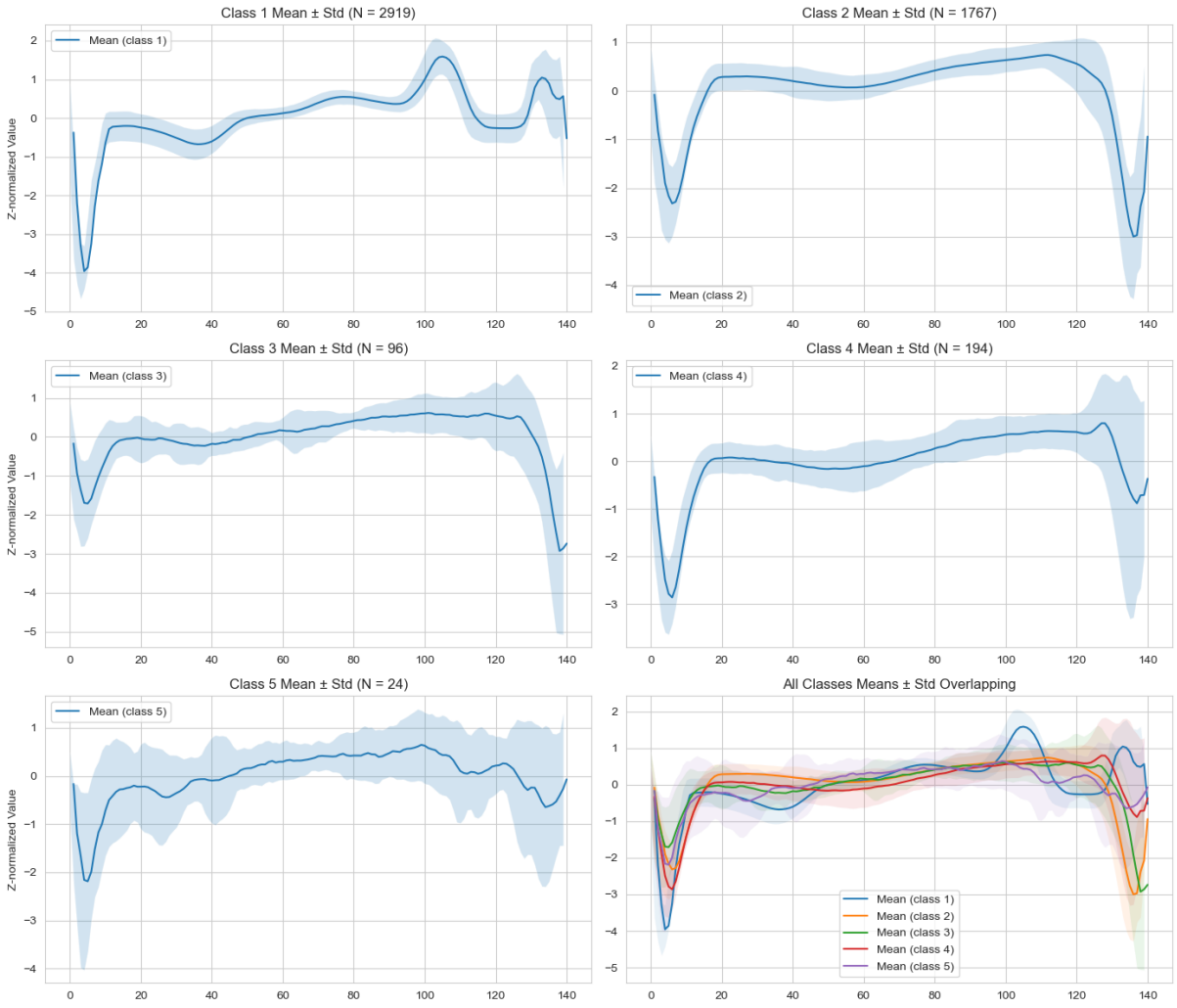

Figure 3. Mean curves and standard deviation bands

2.3 Data Splitting

Although the ECG5000 dataset already provides a test set (4500 samples) and a training set (500 samples), I further split the training set into a training subset and a validation subset (in an 8:2 ratio) for hyperparameter tuning. Since the training set is small and most samples belong to the normal class (Class 1), I used stratified sampling to ensure similar label distributions in both the training and validation subsets. This helps provide a more consistent evaluation of the model's generalization ability and avoids overfitting to the validation set during tuning. The number of samples and class distribution in the split subsets are shown in Table 3, and their distribution is illustrated in Figure 4.

Table 3: Training set and validation set obtained based on stratified sampling partitioning

Dataset	Normal (Class 1)	Abnormal (Class 2-5)
Training Set (N = 400)	233	167
Validation Set (N = 100)	59	41

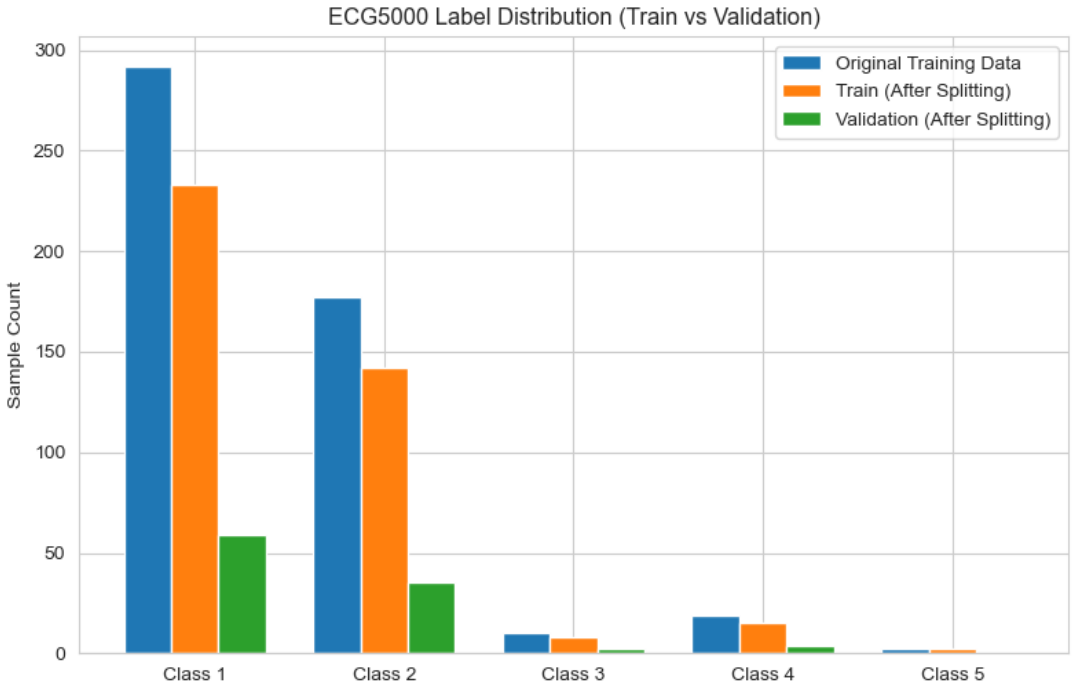

Figure 4. Class distribution before and after splitting

3 Methodology

3.1 Cumulative Sum Control Chart (CUSUM)

3.1.1 Classic CUSUM

The Cumulative Sum Control Chart (CUSUM) is a classic tool in statistical process control that has been widely used and developed over time. Although originally proposed for solving change point detection (CPD) problems, CUSUM can also be applied in anomaly detection (AD) settings, since change points in time series are often related to anomalies [15, 16]. Proposed by E.S. Page (1954), CUSUM was initially designed for industrial quality control, particularly to detect mean shifts in the production process [12]. The core idea of the algorithm is to accumulate the deviation between the observed value and the target value to identify potential changes.

Assuming the observed value x_t follows a normal distribution $N(\mu_0, \sigma^2)$ under the under the null hypothesis, the CUSUM statistics are defined as follows:

$$\begin{cases} \text{Positive CUSUM: } S_t^+ = \max\left(0, S_{t-1}^+ + (x_t - \mu_0 - k)\right) \\ \text{Negative CUSUM: } S_t^- = \max\left(0, S_{t-1}^- + (\mu_0 - x_t - k)\right) \end{cases} \tag{2}$$

An alarm is triggered when either S_t^+ or S_t^- exceeds a predefined threshold h, indicating that a change point is likely to have occurred. The threshold h is typically set to 4 to 5 times the standard deviation.

The parameter k in Equation (2) serves as a reference value to adjust detection sensitivity. For the positive CUSUM S_t^+, if $x_t - \mu_0 > k$, it indicates that the deviation of x_t from the normal mean μ_0 is significant and should be accumulated; otherwise, the deviation is considered negligible. The same logic applies to the negative CUSUM S_t^-. In this way, k controls the sensitivity of the chart to mean shifts: a smaller k increases sensitivity but also raises the false alarm rate. In industrial control applications, a common empirical formula for k is $k = \delta/2$, where δ represents the expected mean shift (expressed in units of σ). When the observation sequence is standardized, (i.e., $\mu_0 = 0, \sigma = 1$), detecting a shift of 1 standard deviation corresponds to $k = 0.5$.

The original definition of CUSUM focuses on detecting mean shifts. While it is easy to understand and implement, it cannot be directly applied to other types of shifts and lacks a solid theoretical foundation. As research in statistical process control and signal detection advanced, scholars began elevating control charts from empirical tools to a framework based on stochastic process theory. In 1971, G. Lorden formally proposed a likelihood ratio-based version of the CUSUM algorithm [13], defined as:

$$S_t = \max\left(0, S_t + \log\frac{f_1(x_t)}{f_0(x_t)}\right) \tag{3}$$

Here, $f_0(x_t)$ and $f_1(x_t)$ are the probability density functions under hypotheses H_0 (no change) and H_1 (change), respectively. An alarm is triggered when $S_t > h$ for a given threshold h. The work of G. Lorden (1971) and Moustakides (1986) demonstrated that, when the distributions are known, the likelihood ratio-based CUSUM method is theoretically optimal in minimizing detection delay [13, 14].

3.1.2 Exponentially Weighted Sliding Window CUSUM (EWSW-CUSUM)

The classic CUSUM method is highly interpretable and computationally efficient, but it also has certain limitations in practice. Page's CUSUM assumes that the data follow a normal distribution, while the likelihood ratio-based CUSUM requires prior knowledge of the distributions before and after the change. Both approaches also assume that the observations are independently and identically distributed (i.i.d.). When these assumptions are violated, directly applying the classic CUSUM may lead to increased false alarm rates or delayed detection. In such cases, nonparametric or adaptive variants of CUSUM become more valuable. For example, Ejaz Ahmed et al. (2008) proposed a dynamic sliding window CUSUM to detect traffic bursts in network data, and Veronica et al. (2010) developed a percentile-based nonparametric CUSUM algorithm [28, 29].

As discussed in Section 2, ECG5000 is a non-stationary time series. Inspired by previous studies, I decided to adopt a sliding window combined with exponential weighting to implement a CUSUM variant capable of dynamically capturing changes—Exponentially Weighted Sliding Window CUSUM (EWSW-CUSUM). The implementation steps of the algorithm are as follows:

1) Sliding Window Statistics

Given a time series of length T, denoted as $x = \{x_1, x_2, \dots, x_T\}$, we define a reference window of length w at any time point t as:

$$W_t = \{x_{t-w}, x_{t-w+1}, \dots, x_{t-1}\} \tag{4}$$

The mean and standard deviation within the window are calculated as:

$$\mu_t = \frac{1}{w}\sum_{i=t-w}^{t-1} x_i \,, \sigma_t = \sqrt{\frac{1}{w-1}\sum_{i=t-w}^{t-1}(x_i - \mu_t)^2} \tag{5}$$

2) Exponentially Weighted CUSUM Recursion

Based on the sliding window statistics, the bidirectional cumulative sums are defined as:

$$\begin{cases} S_t^+ = \max(0, \lambda \cdot S_{t-1}^+ + (x_t - \mu_t - k)) \\ S_t^- = \max(0, \lambda \cdot S_{t-1}^- + (\mu_t - x_t - k)) \end{cases} \tag{6}$$

The initial values of S_t^+ and S_t^- are set to 0. Here, $\lambda \in (0, 1]$ denotes the exponential weighting factor, which serves to discount historical memory. In non-stationary time series, local fluctuations are common. By adjusting the accumulation using the local window standard deviation μ_t, the model can adapt to local dynamic changes. Therefore, in this CUSUM variant, the sensitivity parameter k should be scaled based on μ_t. On the other hand, the alarm threshold should not fluctuate with local variations. To ensure the robustness of the model, it is preferable to estimate the threshold h based on the overall standard deviation of the sequence, denoted as μ_s. A time point t is marked as an alarm when either $S_t^+ > h$ or $S_t^- > h$.

3) Consecutive Alarm Criterion

If there exists a time interval of length l, denoted as $\{t, t+1, \dots, t+l-1\}$, in which all time points are alarm points, then the entire sequence is labeled as an anomalous sequence.

The design motivation behind this CUSUM variant is as follows. First, the sliding window mechanism is a widely used approach for handling non-stationary sequences. It essentially segments a long, non-stationary sequence into multiple short subsequences, each assumed to be locally stationary. On top of this, I applied exponential weighting. In fact, a class of change-point detection methods based on Exponentially Weighted Moving Average Chart (EWMA Chart) has already been proposed in the literature [43-45]. The standard form is:

$$z_t = \lambda \cdot x_t + (1 - \lambda) \cdot z_{t-1} \tag{7}$$

However, the original purpose of the standard EWMA method is to smooth input data, which is more suitable for detecting gradual drifts or long-term trends. Each sample in the ECG5000 dataset represents a single heartbeat cycle, which is a local signal on a very short time scale. Anomalies in such data typically appear as abrupt changes, which do not align with the gradual variations that EWMA is designed to capture. In contrast, I aim to maintain sensitivity to abrupt changes, which is why I adopted the weighted summation form in Equation (6). This formulation preserves the deviation of the most recent observation while introducing λ to gradually forget the accumulated bias from historical distributions, thereby enhancing the method's adaptability to non-stationary sequences.

To illustrate how the algorithm works, I randomly selected a heartbeat from the ECG5000 dataset and used a plot to demonstrate the changes in the positive and negative cumulative sums when applying the method. When setting the window size to $w = 20$, with parameters $k = 0.5\mu_t$, $h = 5\mu_s$, and using $\lambda = 0.9$ as the historical accumulation weight, the results shown in Figure 5 can be obtained.

The positive CUSUM produces 4 consecutive alarms within the interval $[100, 110]$, while the negative CUSUM triggers 5 consecutive alarms around time point 120. Whether this heartbeat is ultimately labelled as an anomaly depends on the chosen threshold for consecutive alarms. If the threshold is set to any value greater than 5, then neither direction will trigger anomaly labelling.

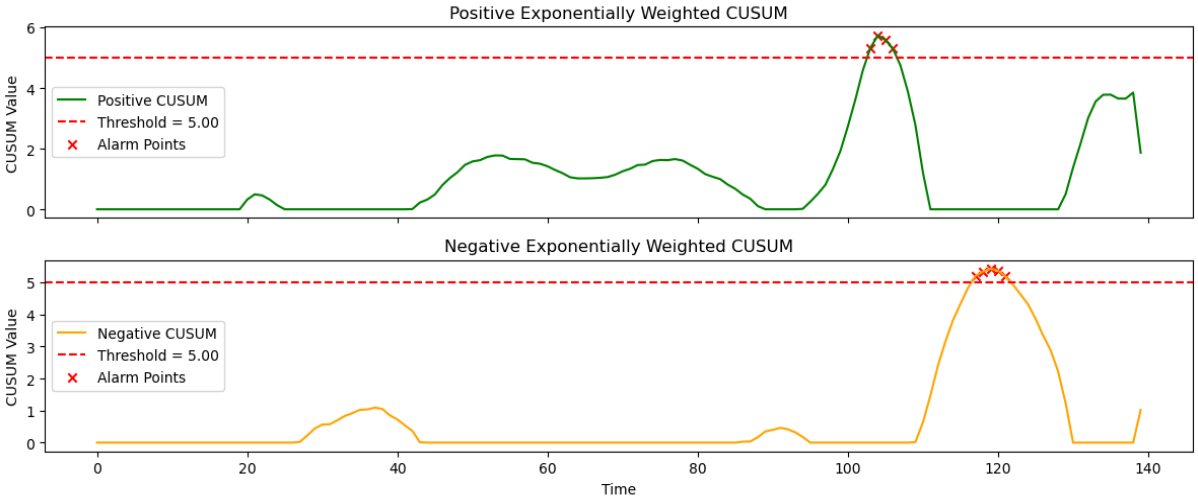

Figure 5. Bidirectional exponentially weighted cumulative sums for all windows

3.2 Copula-Based Outlier Detection (COPOD)

3.2.1 Standard COPOD

COPOD (Copula-Based Outlier Detection) is a relatively new anomaly detection algorithm proposed by Zheng Li et al. in 2020 [30]. It is particularly suitable for identifying anomalies in high-dimensional data and is entirely non-parametric, requiring no assumptions about the data distribution.

The core idea behind COPOD is that, in a multi-dimensional space, a normal point is expected to have moderate values along each dimension. If a data point lies at the extreme ends of all dimensions, it is likely to be an outlier. Based on this intuition, the algorithm turns to the joint distribution of the variables, which leads to the theoretical foundation of COPOD—the Sklar theorem [31]. Given a d-dimensional continuous random variable $(X_1, X_2, \dots, X_d)$ with joint distribution function $F(x_1, x_2, \dots, x_d)$ and marginal distribution functions $F_1(x_1), F_2(x_2), \dots, F_d(x_d)$, there exists a unique copula function $C(\cdot)$ such that:

$$F(x_1, x_2, \dots, x_d) = C\big(F_1(x_1), F_2(x_2), \dots, F_d(x_d)\big) \tag{8}$$

Since the true marginal distribution $F_i(x_i)$ is typically unknown, COPOD estimates it using the empirical cumulative distribution function (ECDF). Given an $n \times d$ sample matrix, where each column represents a feature, the ECDF for the i-th feature X_i is computed as:

$$\hat{F}_i(x_i) = \frac{1}{n}\sum_{j=1}^{n} I\left(x_i^{(j)} \leq x_i\right) \tag{9}$$

Here, $I(\cdot)$ is an indicator function, and $x_i^{(j)}$ denotes the i-th feature value of the j-th sample. The function returns 1 if $x_i^{(j)} \leq x_i$ and 0 otherwise. By applying this estimation to each dimension, the original variable X_i can be transformed into a standard uniform variable $U_j \sim \text{Uniform}(0, 1)$. As a result, the original sample matrix is transformed into an $n \times d$ matrix **U**. After transformation, the i-th feature of the j-th sample becomes:

$$u_i^{(j)} = \hat{F}_i\left(x_i^{(j)}\right), u_i^{(j)} \in [0, 1] \tag{10}$$

Each row of the matrix **U** can then be regarded as a sample drawn from the copula distribution. Using the empirical ranks, the empirical copula function is defined as:

$$\hat{C}(u_1, u_2, \ldots, u_d) = \frac{1}{n}\sum_{j=1}^{n} I\left(u_1^{(j)} \leq u_i, u_2^{(j)} \leq u_2, \ldots, u_d^{(j)} \leq u_d\right) \tag{11}$$

As previously discussed, the core intuition of COPOD is to evaluate how far a data point deviates from the distribution in each of its dimensions. COPOD characterizes this using tail probabilities. Given a test point $\mathbf{x} = (x_1, x_2, \ldots, x_d)$, which is transformed into the copula space as $\mathbf{u} = (u_1, u_2, \ldots, u_d)$, the following definitions apply:

- **Left-tail probability**: $P(\mathbf{U} \leq \mathbf{u})$ represents the proportion of samples in the space for which all dimensions are less than or equal to **u**. Its empirical estimate is:

$$\hat{P}_L = \hat{C}(u_1, u_2, \ldots, u_d) = \frac{1}{n}\sum_{i=1}^{n} I\left(u_1^{(j)} \leq u_i, u_2^{(j)} \leq u_2, \ldots, u_d^{(j)} \leq u_d\right) \tag{12}$$

- **Right-tail probability**: $P(\mathbf{U} \geq \mathbf{u})$ is the proportion of samples where all dimensions are greater than or equal to **u**. To simplify the computation in high-dimensional space, COPOD does not use $\hat{C}(\cdot)$ here, but instead estimates this directly in the original sample space as:

$$\hat{P}_R == \frac{1}{n}\sum_{i=1}^{n} I\left(x_1^{(j)} \geq x_i, x_2^{(j)} \geq x_2, \ldots, x_d^{(j)} \geq x_d\right) \tag{13}$$

To amplify tail probabilities, COPOD takes the negative logarithm and defines the outlier score as:

$$O(x) = \max\left(-\log\left(\hat{P}_L\right), -\log\left(\hat{P}_R\right)\right) \tag{14}$$

After computing the outlier scores for all data points, a threshold can be set based on quantiles. Any point with a score exceeding this threshold is considered an outlier.

The above describes the mathematical derivation of the method based on copulas. In practice, Zheng Li et al. also incorporated additional enhancements, including the use of both-sided tail probabilities and skewness-adjusted tail probabilities, to further improve performance [30].

To further illustrate the execution process of the COPOD method, I conducted the following experiment.

1) A dataset with 100 rows and 2 columns was generated to simulate samples from a two-dimensional random variable. Among them, 95 samples follow a standard bivariate normal distribution, while the remaining 5 are synthetic anomalies. The distribution of these samples is shown in Figure 6.

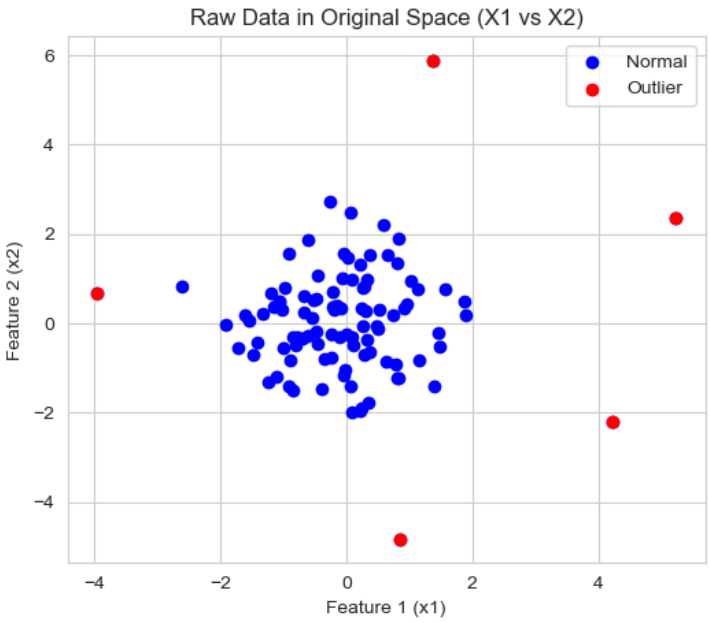

Figure 6. Distribution of a 2D random variable (original sample space)

2) Based on the sample ranks, the ECDFs and frequency histograms of the two features are plotted, as shown in Figure 7.

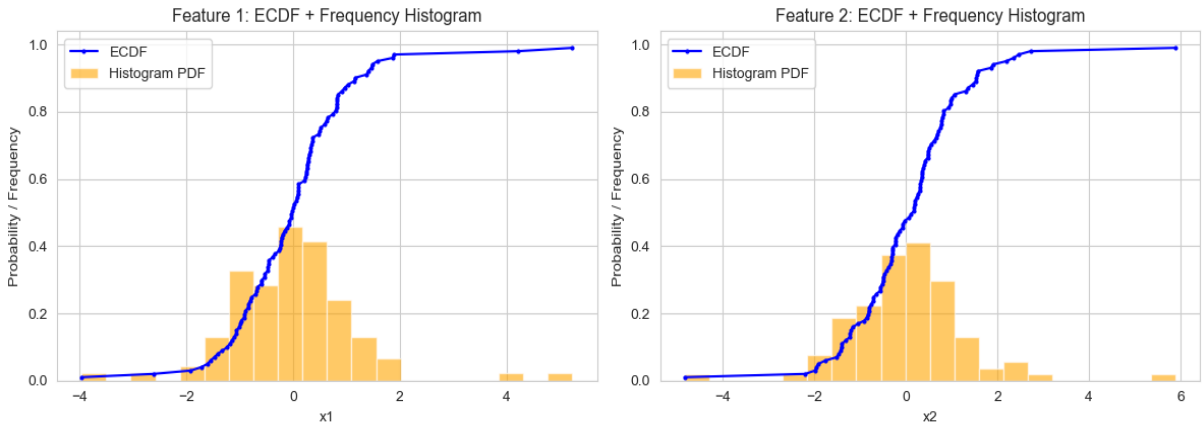

Figure 7. ECDF and frequency histogram of X_1 and X_2

3) Each data point $(x_1^{(j)}, x_2^{(j)})$ is transformed using the ECDF into the copula space as $(u_1^{(j)}, u_2^{(j)})$. The result is shown in Figure 8.

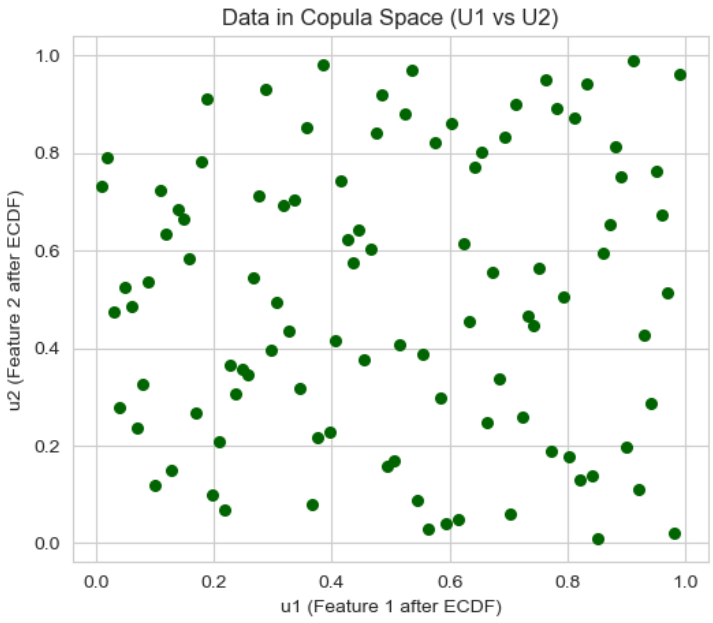

Figure 8. Distribution of 2D random variable (copula space)

4) Finally, an outlier score is computed for each point. Points with extremely high outlier scores are identified as detected outliers. Figure 9 visualizes the outlier scores of all original data points using a heatmap.

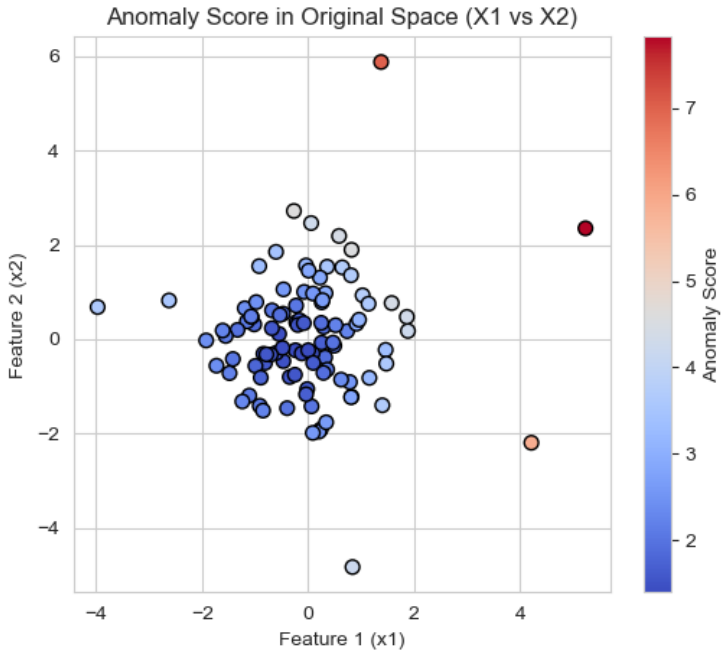

Figure 9. Heatmap of outlier scores for data points

3.2.2 Sliding Window Aggregated COPOD (SWA-COPOD)

Despite its many advantages, I identified a limitation of COPOD during its theoretical analysis—COPOD is a static anomaly detection method, which may fail to effectively capture temporal dependencies or contextual correlations in high-dimensional data. In the ECG5000 dataset, each heartbeat record contains 140 columns, which can be treated as 140 features. Notably, each column corresponds to a specific time point. If the copula distribution is constructed directly on this original feature space, then detecting an anomalous heartbeat indicates that the heartbeat exhibits deviations at several positions within the 140 time points of that interval. While this detection scheme appears intuitive, it overlooks an important clinical fact: anomalous heartbeats are often not isolated events. Many studies and textbooks have pointed out that arrhythmias tend to exhibit continuity, and abnormal heartbeats may appear in clusters or in a sustained fashion [32-34]. This implies that such anomalies are context-dependent. Therefore, considering multiple consecutive heartbeats may help improve anomaly detection performance.

Based on the above analysis, I propose a method called Sliding Window Aggregated COPOD (SWA-COPOD), which flattens and concatenates multiple consecutive samples to form a single sliding window. Compared with the original vectors, each window has a higher dimensionality. Applying COPOD to such high-dimensional representations allows the model to capture correlations across successive heartbeats.

Let the original time series data be represented by a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, where n denotes the number of training samples and d is the feature dimension of each sample. The detailed execution steps of the algorithm are as follows:

1) Sliding Window Construction

Given a window size w and stride s, sliding windows are constructed along the row direction as follows:

$$W = \{\mathbf{X}_{i:i+w-1} | i = 1, 1+s, 1+2s, \dots, n-w+1\} \tag{15}$$

Each window $\mathbf{X}_{i:i+w-1} \in \mathbb{R}^{w \times d}$ consists of w consecutive d-dimensional rows from the original data matrix. Each window is then flattened into a vector:

$$\tilde{\mathbf{x}}^{(k)} = \mathrm{vec}(\mathbf{X}_{i:i+w-1}) \in \mathbb{R}^{w \times d} \tag{16}$$

The resulting collection of windowed samples is:

$$\widetilde{\mathbf{X}} = \left[\tilde{\mathbf{x}}^{(1)}, \tilde{\mathbf{x}}^{(2)}, \dots, \tilde{\mathbf{x}}^{(M)}\right]^{\tau} \in \mathbb{R}^{M \times (w \cdot d)} \tag{17}$$

Here, $M = \left\lfloor \frac{n-w}{s} \right\rfloor + 1$ denotes the maximum number of windows that can be extracted from n rows of original training samples.

2) Constructing the COPOD Model

The COPOD model is built on the feature matrix $\widetilde{\mathbf{X}} \in \mathbb{R}^{M \times D}$, where $D = w \cdot d$. For each feature dimension $i = 1, \dots, D$, the empirical cumulative distribution function (ECDF) is computed as:

$$\hat{F}_i(x_i) = \frac{1}{M} \sum_{j=1}^{M} I\left(\tilde{x}_i^{(j)} \leq x_i\right) \tag{18}$$

Each sample is then mapped to the copula space by:

$$u_i^{(j)} = \hat{F}_i\left(\tilde{x}_i^{(j)}\right) \tag{19}$$

The remaining steps follow the original COPOD procedure to construct the empirical copula distribution $\hat{C}(\cdot)$ (Equation (11)).

3) Window-Level Prediction

For the test sample matrix with n' rows, denoted as $\mathbf{X_{test}} \in \mathbb{R}^{n' \times d}$, the same sliding window operation is applied to obtain windowed test samples:

$$\widetilde{\mathbf{X}}_{\text{test}} = \left[\tilde{\mathbf{x}}_{\mathbf{test}}^{(\mathbf{1})}, \tilde{\mathbf{x}}_{\mathbf{test}}^{(\mathbf{2})}, \ldots, \tilde{\mathbf{x}}_{\mathbf{test}}^{(\boldsymbol{M'})}\right]^{\tau} = \mathbb{R}^{M' \times (w \cdot d)} \tag{20}$$

The original test sequence can be divided into at most $M' = \left\lfloor \frac{n'-w}{s} \right\rfloor + 1$ windows.

For the i-th sample in the original space, the set of windows it contributes to (after transformation) is denoted as:

$$\mathcal{W}_{\text{i}} = \left\{ \tilde{\mathbf{x}}_{\text{test}}^{(k)} \middle| \tilde{\mathbf{x}}_{\text{test}}^{(k)} \text{ belongs to a window containing the } i^{\text{th}} \text{ sample} \right\} \tag{21}$$

Each window is scored using COPOD to obtain an anomaly score s_k. Outlier windows are then identified based on a percentile threshold (contamination rate).

4) Aggregating Window-Level Predictions to Sample-Level

Finally, the prediction results at the window level need to be mapped back to the sample level. This step is highly flexible and can be implemented in different ways. One approach is to aggregate window anomaly scores. For example, the anomaly score of the current sample s_i can be computed as the average score of all windows that cover this sample:

$$s_i = \frac{1}{|\mathcal{W}_{\text{i}}|} \sum_{\tilde{\mathbf{x}}_{\text{test}}^{(k)} \in \mathcal{W}_{\text{i}}} s_k \tag{22}$$

Another approach is to aggregate the window-level classification results by computing the proportion of anomalous windows in $\mathcal{W}_{\text{i}}$. If this proportion exceeds a threshold, the corresponding sample is labeled as anomalous.

The above provides a formal description of SWA-COPOD. In the original COPOD algorithm, anomaly scores are derived by computing tail probabilities for each sample across all feature dimensions. These scores mainly depend on two factors:

- A particular feature value is significantly large or small in its dimension, i.e., it falls into the tail region of the ECDF (close to 0 or 1)
- The sample exhibits tail deviations in multiple dimensions, resulting in an overall higher anomaly score

Based on these two points, the expected performance improvement of SWA-COPOD can be attributed to the effects of “anomaly diffusion” and “anomaly amplification”.

- **Anomaly Diffusion Effect (Training Phase)**

  If an outlier exists in the training set, it may be included in multiple windows across different dimensions due to the sliding window mechanism. When constructing the ECDF for the affected dimensions, the tail regions may be stretched. This smoothing of the tails can enhance model robustness, reducing the chance of false alarms caused by noise or moderate deviations during testing.

  Example: Suppose we collect 200 observations from a specific dimension of a high-dimensional dataset, among which 195 values follow a standard normal distribution, and the remaining 5 are outliers.

  We first construct an ECDF from all 200 observations, then construct a baseline ECDF by replacing the 5 outliers with normal values. As illustrated in Figure 10, the grey dashed line (with probability = 0.95) denotes the threshold for the tail region. The red curve (with outliers) shows a long right tail, while the tail of the blue curve (with no outliers) is steeper and shorter. If the observation falls near 2, its probability under the red curve is around 0.95, meaning that only extremely deviated values (within $[4, +\infty]$) would result in large tail probabilities. In contrast, under the blue curve, even moderately deviated values (within $[2, 3]$) can lead to high probability values.

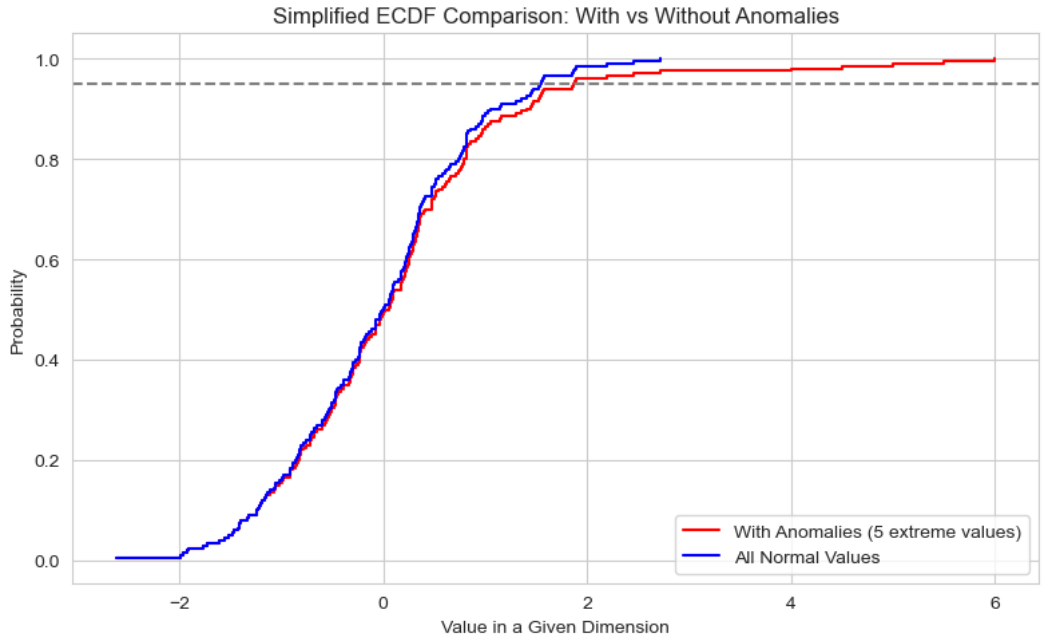

Figure 10. Impact of anomalies on the tail structure of the ECDF

- **Anomaly Amplification Effect (Testing Phase)**

  If a group of adjacent anomalous heartbeats exists in the test set, their anomalous segments may become aligned within the same feature dimension due to the overlapping nature of sliding windows. This causes the affected dimension to receive high anomaly scores repeatedly, thereby increasing the anomaly score of the window. When window-level scores are later aggregated back to the sample level (per heartbeat), the resulting sample-level anomaly score also increases. In this way, the shared structure among abnormal heartbeats can amplify the anomalies and enhance the model's detection capability.

  Example: Consider a test set of shape 30×10, where each sequence contains 30 samples with 10 features. Let $S_i, i \in [1, 30]$ denote the i-th test sample. All dimensions follow a standard normal distribution, except that the 7-th dimension of samples S_{21} to S_{23} are assigned extreme values to simulate a shared anomaly pattern.

  Using SWA-COPOD with a window size of 5 and stride 1, both the training and test sets are transformed into high-dimensional datasets of shape 26×50 ($\left\lfloor \frac{30-5}{1} \right\rfloor + 1 = 26, 5 \times 10 = 10$). Each window contains a pair of samples in the form of $[S_1, \dots, S_5], [S_2, \dots, S_6], [S_3, \dots, S_7], \dots, [S_{26}, \dots, S_{30}]$. After transformation, the

anomalous value in S_{21} first appears in the 47-th feature of window 17 ($\mathcal{W}_{17} = [S_{17}, \ldots, S_{21}]$). As the window slides, the anomalies in neighbouring samples (dark-coloured blocks) become aligned at dimensions 47, 37, 27, 17, and 7 in subsequent windows. As a result, the anomaly scores at these dimensions are amplified (see Figure 11).

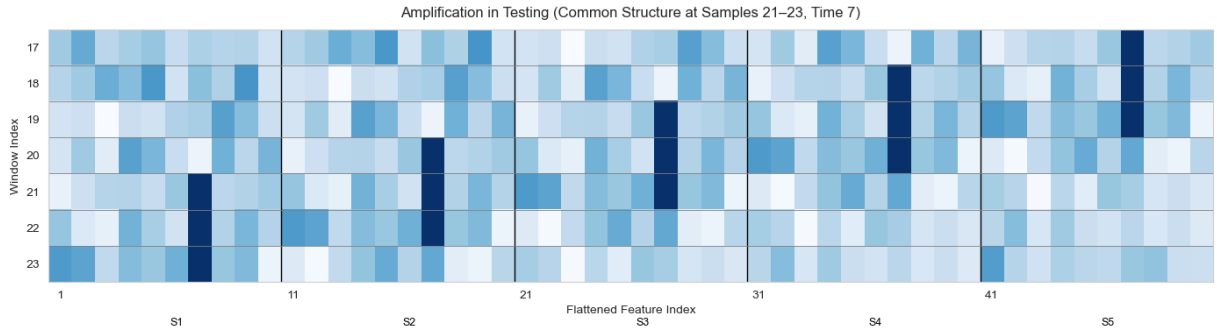

Figure 11. Aligned anomaly dimensions across neighbouring samples within the same window feature

3.3 Isolation Forest (iForest)

3.3.1 Standard iForest

The Isolation Forest (iForest) algorithm, proposed by Fei Tony Liu et al. in 2008, is a widely used method for anomaly detection [35]. Unlike earlier approaches, which typically relied on assumptions about the data distribution or required distance- or density-based metrics, iForest introduces a novel mechanism based on the idea of random partitioning and isolation. It offers strong performance with linear time complexity, making it well-suited for anomaly detection in high-dimensional datasets.

The core intuition behind iForest is that anomalies are “isolated” and “different” from the rest of the data. Here, “different” typically means deviating—being either unusually large or small—which allows anomalous values to be separated from the rest with fewer random partitioning operations. An iForest consists of a collection of isolation trees (iTrees). To improve isolation efficiency, the algorithm applies sub-sampling: it randomly selects ψ samples from a dataset of n points to build a single iTree. Given a sample matrix of size $n \times d$, the construction of an iTree proceeds as follows: randomly select a feature dimension i from the d available features and then choose a split point p along this dimension. The split value p is drawn uniformly at

random from the range between the minimum and maximum values of the current subset along dimension i, that is:

$$p \sim \text{Uniform}(\min_i, \max_i) \tag{23}$$

Let $x_i^{(j)}$ denote the i-th feature value of the j-th sample. The dataset is recursively split based on the condition $x_i^{(j)} < p$ and $x_i^{(j)} > p$, dividing the ψ samples into left and right child nodes. The recursion terminates under one of the following three conditions:

- The current node contains only one sample and can no longer be split.
- All samples in the node have identical values. For high-dimensional data, this requires equality across all feature dimensions, i.e., for any two samples in the node, $x_i^{(j)} = x_i^{(k)}, i \in \{1, 2, \dots, d\}$.
- The tree has reached the maximum depth limit, typically set to $\log_2(\psi)$.

Once an iTree is built, the path length of a sample x, denoted as $h(x)$, is defined as the number of edges from the root node to the leaf node that contains x—that is, the number of splits encountered by x before recursion terminates. According to Donald E. Knuth (1973) [36], the average depth of a node in a random binary search tree (BST) with n nodes is:

$$E[D(x)] \approx 2H(n) - 2 \tag{24}$$

Here, $H(n)$ denotes the n-th harmonic number, which is calculated as follows (γ is the Euler's constant, approximately 0.577):

$$H(n) = \sum_{k=1}^{n} \frac{1}{k} \approx \ln n + \gamma \tag{25}$$

In Knuth's formulation (Equation (24)), the random variable $D(x)$ represents the number of edges from the root to node x, which could also be an internal node. In the case of an iTree, what we care more about is the stopping point of splitting—that is, the number of edges from the root to a leaf node $h(x)$. Since a leaf in an iTree may contain multiple samples, the path length cannot be estimated solely by Knuth's formula and must be adjusted based on the size of the leaf node. To this end, the

authors proposed a correction term $c(n)$, which denotes the average path length of a subtree containing n samples:

$$c(n) = \begin{cases} 2H(n-1) - \dfrac{2(n-1)}{n}, & n > 1 \\ 0, & n = 1 \end{cases} \tag{26}$$

Thus, the path length of a sample x in a single iTree can be expressed as:

$$h(x) = e(x) + c(n_l) \tag{27}$$

Here, $e(x)$ is the number of edges from the root to the leaf node, and n_l is the number of samples contained in that leaf. To enhance the algorithm's robustness, iForest builds t iTrees and computes the average path length over them as the expected path length of x:

$$h(x) = \frac{1}{t}\sum_{i=1}^{t} h_i(x) \approx E[h(x)] \tag{28}$$

By the law of large numbers, when t is sufficiently large, $h(x) \to E[h(x)]$.

Like COPOD, iForest ultimately needs to quantify the likelihood that a sample is an anomaly by computing an anomaly score. Since the expected path lengths of different samples may vary significantly, iForest normalizes the path length into an anomaly score:

$$s(x, \psi) = 2^{-\frac{E[h(x)]}{c(\psi)}} \tag{29}$$

Here, $c(\psi)$ is the normalization factor, representing the average path length of a subtree with size ψ. It can be calculated using Equation (26). After normalization, we have $s(x, \psi) \in [0, 1]$. According to the anomaly scoring rule:

- If $E[h(x)] \ll c(\psi)$, then $s(x, \psi) \to 1$, meaning it takes fewer splits to isolate x, which is likely an anomaly.
- If $E[h(x)] \gg c(\psi)$, then $s(x, \psi) \to 0$, indicating many splits are needed to isolate x, which is likely normal.
- If $E[h(x)] \approx c(\psi)$, then $s(x, \psi) \approx 0.5$, suggesting the anomaly status of x is uncertain.

To better illustrate the execution process of iForest, I visualized the data splitting steps, as shown in Figure 12. Consider a dataset of size 30×2, where each sample has two feature dimensions X and Y. Among them, 27 samples are drawn from a bivariate normal distribution (blue dots), while the remaining 3 are anomalies (red dots) that significantly deviate from the center. For simplicity, I did not apply sub-sampling; instead, a single iTree was built using the entire dataset with $\psi = 30$, and the split termination conditions remain the same as in the original iForest algorithm.

In Figure 12, vertical dashed lines indicate splits on feature X with a randomly selected split value p_X within its range; horizontal dashed lines indicate splits on feature Y with a randomly chosen value p_Y. After the tree is built, each anomaly is separated in no more than 3 splits, while the average path length for normal points reaches 8.81. This result reconfirms the theoretical intuition behind iForest: anomalies are likely located on the edge or isolated, thus requiring only short paths to be separated.

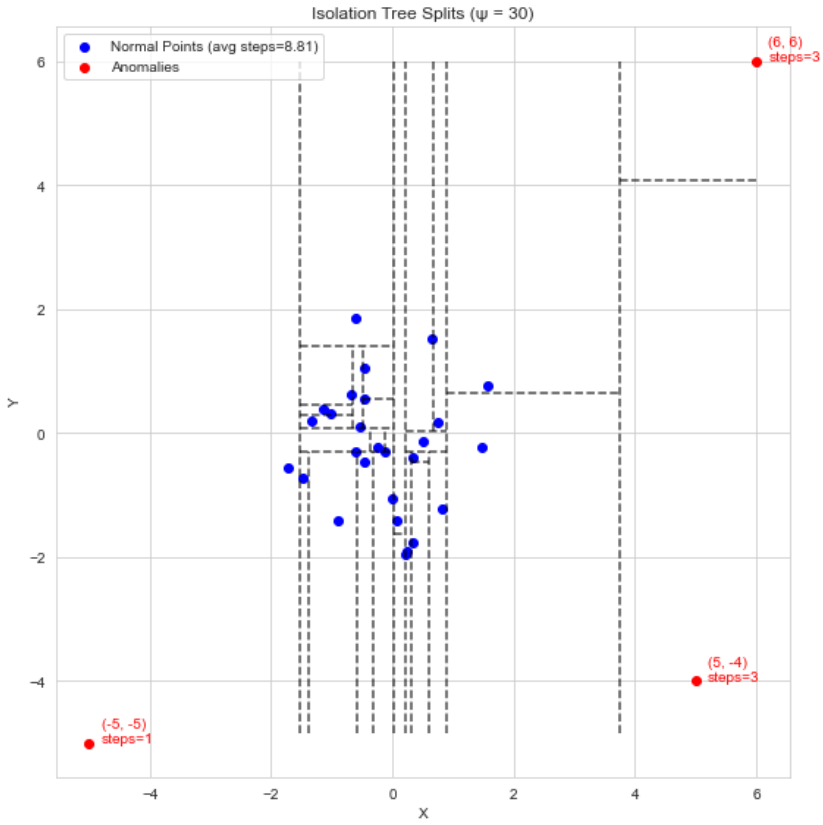

Figure 12. Dataset split as a single iTree ($\psi = 30$)

Figure 13 shows the anomaly score heatmap of all points in the iTree. As observed, all anomalous points have anomaly scores above 0.7, indicating strong abnormality, while most normal points near the centre experience more splits and are shown in blue, reflecting lower anomaly scores.

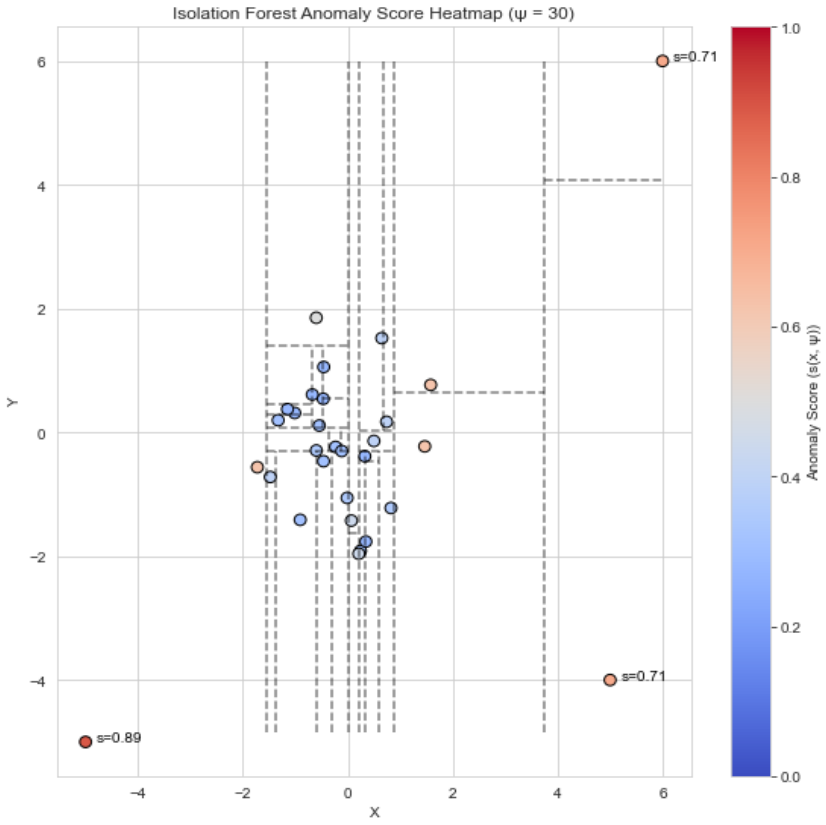

Figure 13. Anomaly score heatmap after dataset splitting ($\psi = 30$)

3.3.2 PCA-iForest

Fei Tony Liu et al. have pointed out that the iForest algorithm also suffers from the curse of dimensionality [35]. From the execution process of iForest, it is clear that the algorithm heavily relies on the selection of dimensions during random splitting. When the sample space has too many dimensions, it may lead to:

- Many redundant features, which increase the path length in the iTree during partitioning
- Greater susceptibility to noise dimensions
- Increased computational overhead for the model

When conducting exploratory analysis on the full ECG5000 dataset, I reached a preliminary conclusion: except for the intervals [0, 20] and [100, 140], where the means and standard deviations of normal and anomalous classes differ significantly, the differences in other time segments are relatively small. To further support this conclusion, I plotted the variance at each time point (Figure 14):

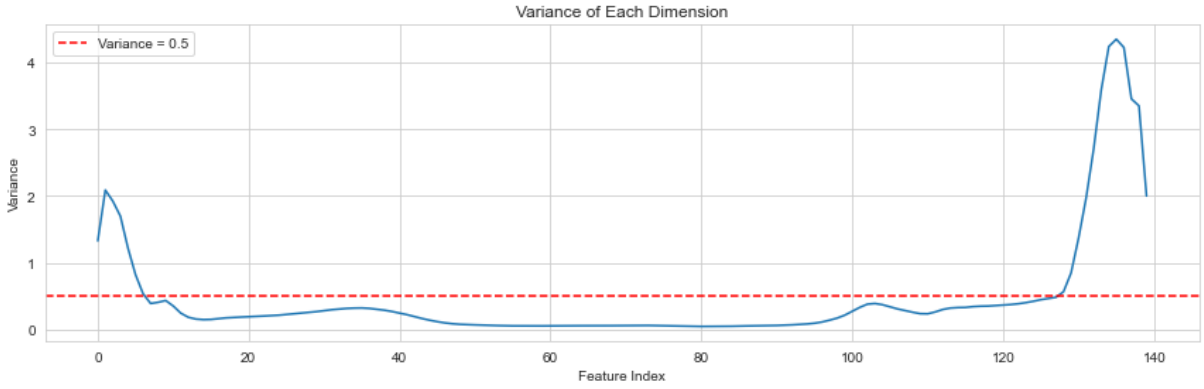

Figure 14. ECG5000 variance per time point

The plotted results closely align with my earlier observation: within the interval [20, 100], the variance is relatively flat, with an average below 0.5, and the variance of several dimensions approaches zero. This indicates that in this range, the difference between normal and abnormal heartbeats is minimal. If iForest is applied directly to the ECG5000 dataset, it is likely to suffer from the curse of dimensionality caused by these features. Therefore, I decided to introduce Principal Component Analysis (PCA) as an optimization step before applying iForest.

The theoretical details of PCA are beyond the scope of this paper; here I only summarize the main steps:

- **Compute the covariance matrix:** Calculate the covariance matrix of the centered feature values. The eigenvectors of the matrix represent the directions of the principal components, and the eigenvalues indicate the variance along those directions.
- **Select principal components:** Rank the eigenvalues in descending order and select the top k eigenvectors as the principal component directions.
- **Linear transformation:** Transform the centered data linearly into a lower-dimensional space based on the selected principal component directions.

Essentially, a principal component is a weighted combination of original features. It is chosen to capture the maximum variance in the data when samples are projected onto this direction. By using these directions as new axes and projecting the original data onto them, dimensionality reduction can be achieved.

Figure 15 shows the explained variance of the first 20 principal components obtained by applying PCA to the full ECG5000 dataset. The red curve represents the cumulative explained variance, which intersects the horizontal 90% threshold at the 10th principal component. This indicates that the first 10 components can explain more than 90% of the total variance. Therefore, when applying iForest later, it is possible to reduce the dimensionality from 140 to 10 while retaining about 90% of the original information. Although the figure is based on the full dataset, in practice, PCA should be fitted using only the training set. The resulting principal component directions should then be used to project the validation and test sets. Applying PCA directly to the entire dataset would lead to information leakage.

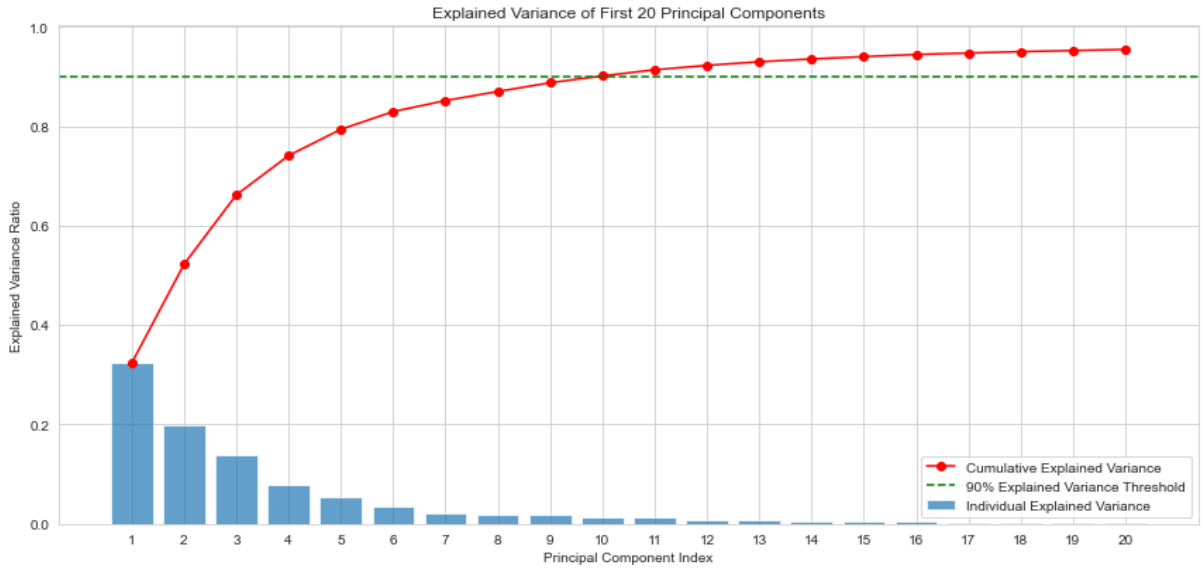

Figure 15. The first 20 principal components of ECG5000 with cumulative explained variance.

3.4 One-Dimensional Convolutional Neural Network (1D-CNN)

At the end of the last century, Yanne LeCun and colleagues developed LeNet-5 (1998), which was the first complete and systematic neural network architecture [37]. It was successfully applied to handwritten character recognition tasks and is widely regarded as the origin of convolutional neural networks (CNNs). In recent years, with advances in computing power, particularly the rise of GPUs, CNNs have been widely

adopted in fields such as computer vision and natural language processing [38, 39]. Traditional CNNs typically take two- or three-dimensional data as input. However, as CNNs have expanded into new domains, an increasing number of scenarios involve one-dimensional input data. As a result, a method known as One-Dimensional Convolutional Neural Network (1D-CNN) has been developed and widely adopted [40-42].

The modeling process of 1D-CNN is as follows:

1) **Input Layer:** A single-channel sequence of length T, typically representing time series, signal data, etc.

2) **Convolutional Layer**: This layer uses F one-dimensional convolutional kernels of width k. Each kernel works like a filter. It slides over the input sequence to compute weighted sums and extract features. At each step, it multiplies its k weights with k input values in the current window, sums the result, and adds a bias term b to increase the flexibility.

   Let the kernel be $\mathbf{w} = [w_1, w_2, \dots, w_k]$ and the stride be s. Then the convolution can be written as:

$$y_i = \sum_{j=1}^{k} w_j \cdot x_{(i-1)*s+j} + b\,, i = 1, 2, \dots, \left\lfloor \frac{T-k}{s} \right\rfloor + 1 \tag{30}$$

   The output width of each feature map is $\left\lfloor \frac{T-k}{s} \right\rfloor + 1$, and there are F such feature maps.

3) **Activation Function**: The convolutional layer performs only linear transformations, which limits its ability to represent complex patterns. To address this, a nonlinear activation function is applied. The most common one is the ReLU function, defined as $f(x) = \max(0, x)$. It processes the output feature map as follows:

$$z_i = \sigma(y_i) = \mathrm{ReLU}(\mathrm{y_i}) = \max(0, y_i) \tag{31}$$

4) **Pooling Layer**: The pooling layer reduces the number of parameters and helps prevent overfitting. It also lowers the computational cost of the following layers. A common method is max pooling, where a sliding window moves across the feature

map and outputs the maximum value in each window. These outputs together form a new feature map.

A complete CNN layer consists of a convolutional layer, an activation function, and a pooling layer. It represents one round of feature extraction. A CNN model—whether 1D or 2D—can stack multiple such layers to extract higher-level and more abstract features.

5) **Fully Connected Layer**: In a 1D-CNN, the feature maps output from convolutional layers have a two-dimensional structure. To use them in a fully connected layer, they must first be flattened into a one-dimensional vector.

A fully connected layer consists of several neurons. Each neuron takes the features from the CNN layer as input, computes a weighted sum, adds a bias term, and applies an activation function to introduce non-linear relationships between features. The number of neurons determines the dimensionality of the output vector.

Assume there are m neurons in the current layer. Given an input vector $\mathbf{x} = [x_1, x_2, \dots, x_n] \in \mathbb{R}^n$, the weight vector of the j-th neuron is $\mathbf{w}^{(j)} = [w_1^{(j)}, w_2^{(j)}, \dots, w_n^{(j)}]$ and the bias is $b^{(j)}$. The operation of the fully connected layer can be expressed as:

$$y_i = \sigma\left(\sum_{i=1}^{n} w_i^{(j)} x_i + b^{(j)}\right), j = 1,2, \dots, m \tag{32}$$

The output is a high-level feature vector $\mathbf{y} = [y_1, y_2, \dots, y_m]$, as illustrated in Figure 16.

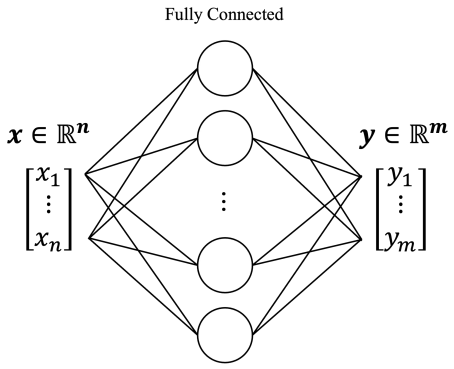

Figure 16. Fully connected layer: inputs to outputs

6) **Output Layer**: For a binary classification task, the output layer can use a single neuron with a Sigmoid function to map the high-level features **y** from the previous layer to a classification probability. During training, this probability is passed into a loss function to guide backpropagation and update the network parameters. During testing, the probability serves as the prediction result, and a predefined threshold can be used to determine the final class label.

To further illustrate the 1D-CNN approach, I selected a sample from the ECG5000 dataset to demonstrate the training process of a 1D-CNN model with a single CNN layer. The entire process is shown in Figure 17.

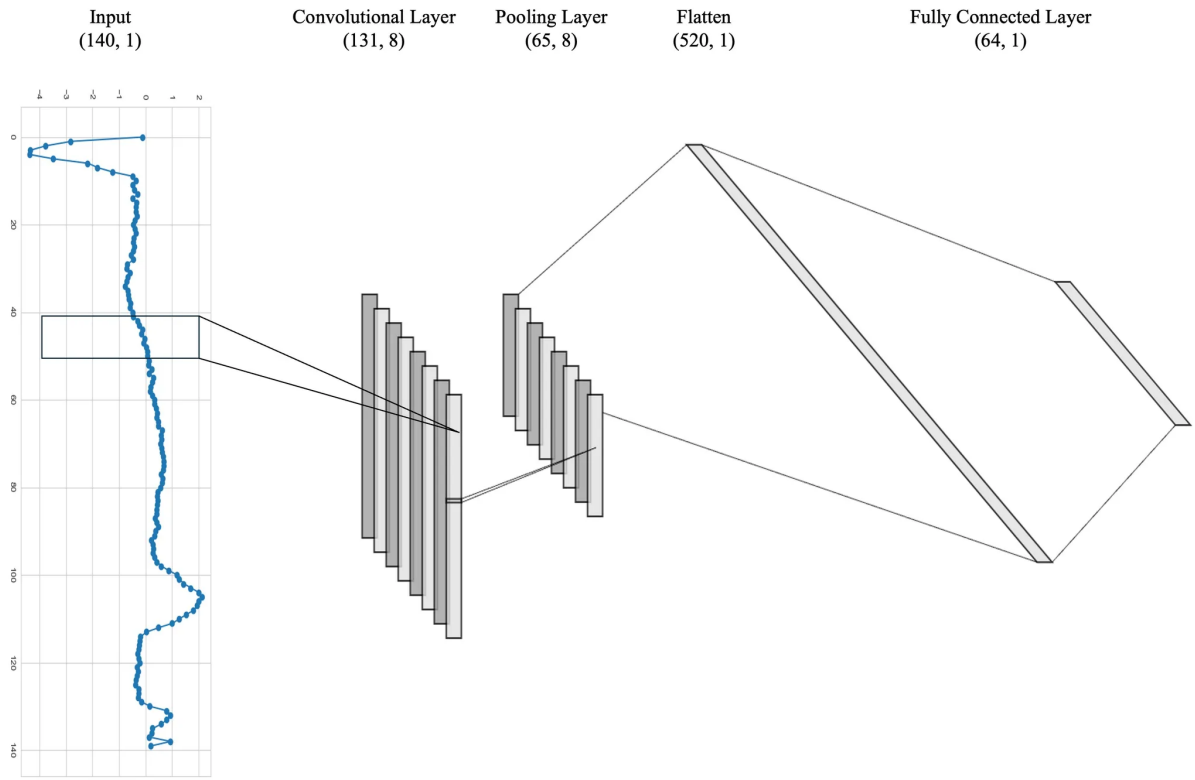

Figure 17. Training a 1D-CNN using ECG5000 with a single CNN layer

1) **Input Layer**: Each sample is a univariate time series of length 140, so the input shape is (140, 1).

2) **Convolutional Layer**: In this layer, 8 convolutional filters of size 10 are applied with a stride of 1. Each filter acts as a sliding window that moves along the time axis, observing 10 consecutive time steps at a time. Each filter multiplies its 10 weights with the 10 input values, sums the results, and adds a bias. For an input of

length 140, a single filter produces an output of length 131 ($\frac{140-10}{1} + 1 = 131$). With 8 filters, the output shape becomes $(131, 8)$.

3) **Activation Function**: A ReLU function is applied to each value, keeping the feature map shape as $(131, 8)$.

4) **Pooling Layer**: Using a max pooling window of size 2 with no overlap, the length of each feature map is reduced to 65 ($\left\lfloor\frac{131-2}{2}\right\rfloor + 1 = 65$), resulting in an output shape of $(65, 8)$.

5) **Fully Connected Layer**: After flattening, the total number of features becomes 520 ($65 \times 8 = 520$), so the input shape becomes $(520, 1)$. This is passed to a fully connected layer. If the layer has 64 neurons, each sample will produce a new vector of shape $(64, 1)$ as its high-level representation.

6) **Output Layer**: A Sigmoid function is applied to the 64-dimensional feature vector to produce a predicted probability.

4 Experiments and Results

4.1 Evaluation Metrics

The task of modeling and prediction on the ECG5000 dataset is essentially a binary classification problem. The goal is to identify abnormal heartbeats (Class 2–5), so abnormal heartbeats are defined as Positive, while normal heartbeats (Class 1) are considered Negative. Based on this setup, we can construct the confusion matrix shown in Table 4.

Table 4: Binary confusion matrix

	Predicted Abnormal	**Predicted Normal**
Actual Abnormal	TP (True Positive)	FP (False Negative)
Actual Normal	FP (False Positive)	TN (True Negative)

Based on the confusion matrix, we can evaluate the model's prediction performance using the following metric:

- **Accuracy**: Accuracy measures the proportion of correctly predicted samples (true positives and true negatives) among all samples.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{33}$$

- **Precision**: Precision measures the proportion of correctly predicted positive samples among all predicted positives. A high precision indicates that the model has a low false positive rate.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{34}$$

- **Recall**: Recall measures how many actual positives are correctly identified by the model. A high recall means the model can capture most of the abnormal data.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{35}$$

- **F1 Score**: As the harmonic mean of precision and recall, the F1 score provides a balanced measure of the model's predictive performance.

$$\mathrm{F1} = \frac{2 * \mathrm{Precision} * \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \tag{36}$$

- **ROC Curve**: The Receiver Operating Characteristic (ROC) curve illustrates the relationship between the True Positive Rate (TPR) and the False Positive Rate (FPR) at different classification thresholds based on anomaly scores. It provides a comprehensive view of a model's performance across different decision boundaries.

$$\mathrm{TPR} = \frac{TP}{TP + FN}, \mathrm{FPR} = \frac{FP}{FP + TN} \tag{37}$$

The area under the ROC curve (AUC) measures the overall discriminative ability of the model. Its value lies within the interval $[0, 1]$. A higher AUC indicates better ranking performance, meaning the model is more likely to assign higher scores to positive samples than to negative ones. When AUC = 1, it represents a perfect classifier, while AUC = 0.5 suggests the model performs no better than random guessing.

4.2 Hyperparameter Tuning

To improve the performance of different models during the prediction phase, I performed hyperparameter tuning using the validation set. I chose F1-score as the evaluation metric for hyperparameter optimization, since it considers both precision and recall and provides a balanced view of false positives and false negatives. Tables 5–8 present the parameter settings and optimal configurations for the four methods.

Table 5: EWSW-CUSUM

Parameter	Meaning	Search Space	Optimal Setting
window_size	Window size	[10, 20, 50]	20
k	Sensitivity parameter	[$0.3\mu_t$, $0.5\mu_t$, $0.7\mu_t$]	$0.3\mu_t$
h	Alarm threshold	[$3\mu_s$, $5\mu_s$, $7\mu_s$]	$7\mu_s$
lam	Weight of historical accumulation	[0.8, 0.9, 0.95]	0.8
alarm_lens	Consecutive alarm threshold	[3, 5, 7]	3

Table 6: SWA-COPOD

Parameter	Meaning	Search Space	Optimal Setting
window_size	Window size	[20, 30, 40]	30
stride	Window stride	[2, 4, 6]	4
contamination	Assumed anomaly proportion	[0.1, 0.15, 0.2]	0.1
ratio_threshold	Abnormal window ratio threshold	[0.2, 0.4, 0.6]	0.2

Table 7: PCA-iForest

Parameter	Meaning	Search Space	Optimal Setting
pca_variance_ratio	Proportion of original data variance retained by PCA	[0.9, 0.95, 0.99]	0.95
estimators	Number of iTrees in iForest	[50, 100, 150]	50
contamination	Assumed anomaly proportion	[0.01, 0.05, 0.1, 0.4]	0.4

Table 8: 1D-CNN

Parameter	Meaning	Search Space	Optimal Setting
conv1_filters	Number of filters in the first convolutional layer	[16, 32, 64]	64
conv2_filters	Number of filters in the second convolutional layer	[32, 64, 128]	32
dense_units	Number of neurons in the fully connected layer	[32, 64, 128]	64

4.3 Results and Discussion

After determining the optimal parameters for each model using the validation set, I proceeded to build anomaly detection models on the prediction set. To comprehensively evaluate the effectiveness of the proposed method, I also introduced several classical algorithms as baselines for comparison. The following models were included in the evaluation:

CUSUM-Based

- **EWSW-CUSUM**: The proposed variant of CUSUM designed for non-stationary time series.
- **SW-CUSUM**: A sliding window version of CUSUM used as a baseline to compare against EWSW-CUSUM. The differences lie in: (1) the cumulative sum is computed using the standard CUSUM formulation without applying any forgetting weights to historical data; (2) anomaly labelling is based on single

alarm points rather than requiring consecutive alarms. All other parameters are consistent with EWSW-CUSUM.

COPOD-Based

- **SWA-COPOD**: The proposed variant of COPOD tailored for non-stationary time series.
- **COPOD**: The original nonparametric COPOD algorithm.

iForest-Based

- **PCA-iForest**: iForest implemented after dimensionality reduction using PCA.
- **iForest**: The original iForest algorithm, using the same parameters as PCA-iForest.

Deep Learning

- **1D-CNN:** A convolutional neural network with two CNN layers, using ReLU activation and max pooling.

Table 9: Evaluation of Prediction Results

Method / Metric	Accuracy	Precision	Recall	F1-Score
SW-CUSUM	0.4169	0.4165	1.0000	0.5881
EWSW-CUSUM	0.9527	0.9952	0.8905	0.9400
COPOD	0.6207	0.6673	0.1767	0.2794
SWA-COPOD	0.9669	0.9349	0.9893	0.9613
iForest	0.6178	0.5410	0.5387	0.5399
PCA-iForest	0.6304	0.5504	0.5750	0.5643
1D-CNN	0.9804	0.9972	0.9557	0.9760

Table 9 presents a comprehensive comparison of all methods across four metrics: Accuracy, Precision, Recall, and F1-Score. The differences among them are significant.

First, looking within the same method category, although SW-CUSUM achieves a perfect Recall score of 1.0000, its Precision is only 0.4165 and the F1-Score is just 0.5881. This indicates that while it successfully captures all anomaly points, it also produces many false positives. In contrast, EWSW-CUSUM introduces exponential

weighting and a consecutive alarm mechanism, which effectively reduce the frequent false alarms seen in the standard sliding window method. As a result, its Precision increases dramatically to 0.9952. Although Recall slightly drops to 0.8905, the overall F1-Score reaches 0.9400, significantly outperforming SW-CUSUM. This demonstrates that the two mechanisms not only suppress noise-triggered false alarms effectively but also preserve strong anomaly detection capability.

Compared to the original COPOD algorithm, SWA-COPOD introduces a multi-sample sliding window mechanism. This leads to a modest improvement in Precision, increasing from 0.6673 to 0.9349. However, the improvement in Recall is much more significant—it rises sharply from 0.1767 to 0.9893. As a result, the overall F1-Score rises to 0.9613, more than three times higher than the original COPOD score of 0.2794. This shows that SWA-COPOD effectively reduces COPOD's over-sensitivity to isolated points. At the same time, it can better capture contextual structural anomalies. In other words, the "anomaly diffusion effect" and "anomaly amplification effect" that I proposed in the methodology section are well validated here.

The prediction results of iForest and PCA-iForest show that introducing PCA brings only a slight performance improvement. This is closely related to the internal mechanism of iForest. iForest relies on sample density in the feature space to detect anomalies. While PCA helps avoid the curse of dimensionality by compressing redundant features, it may also remove key local patterns. In ECG5000, anomalies often appear as subtle local waveform changes. So, PCA's limited benefit for iForest is understandable.

1D-CNN gives the best results, with an F1-Score of 0.9760. Precision and Recall are also high (0.9972 and 0.9557), showing good sensitivity and noise tolerance. But unlike statistical methods, 1D-CNN needs a lot of parameters and training time, so the cost is much higher. In comparison, the proposed methods, EWSW-CUSUM and SWA-COPOD, only depend on statistical computations, yet they achieve detection results close to 1D-CNN. They are especially practical in scenarios where real-time performance and computational resources are limited.

The following describes the ROC curves of the three improved algorithms and the 1D-CNN model. Since ROC evaluation requires each model to output an anomaly score,

and not all methods naturally produce such a score, I define the anomaly scores as follows:

- **EWSW-CUSUM**: The maximum cumulative sum within a sample is used as the anomaly score for that interval.
- **SWA-COPOD**: The anomaly score is defined as the proportion of anomalous windows among all windows that include the given sample.
- **PCA-iForest**: The anomaly score is directly taken from the iForest output.
- **1D-CNN**: The anomaly score is directly taken from the 1D-CNN output.

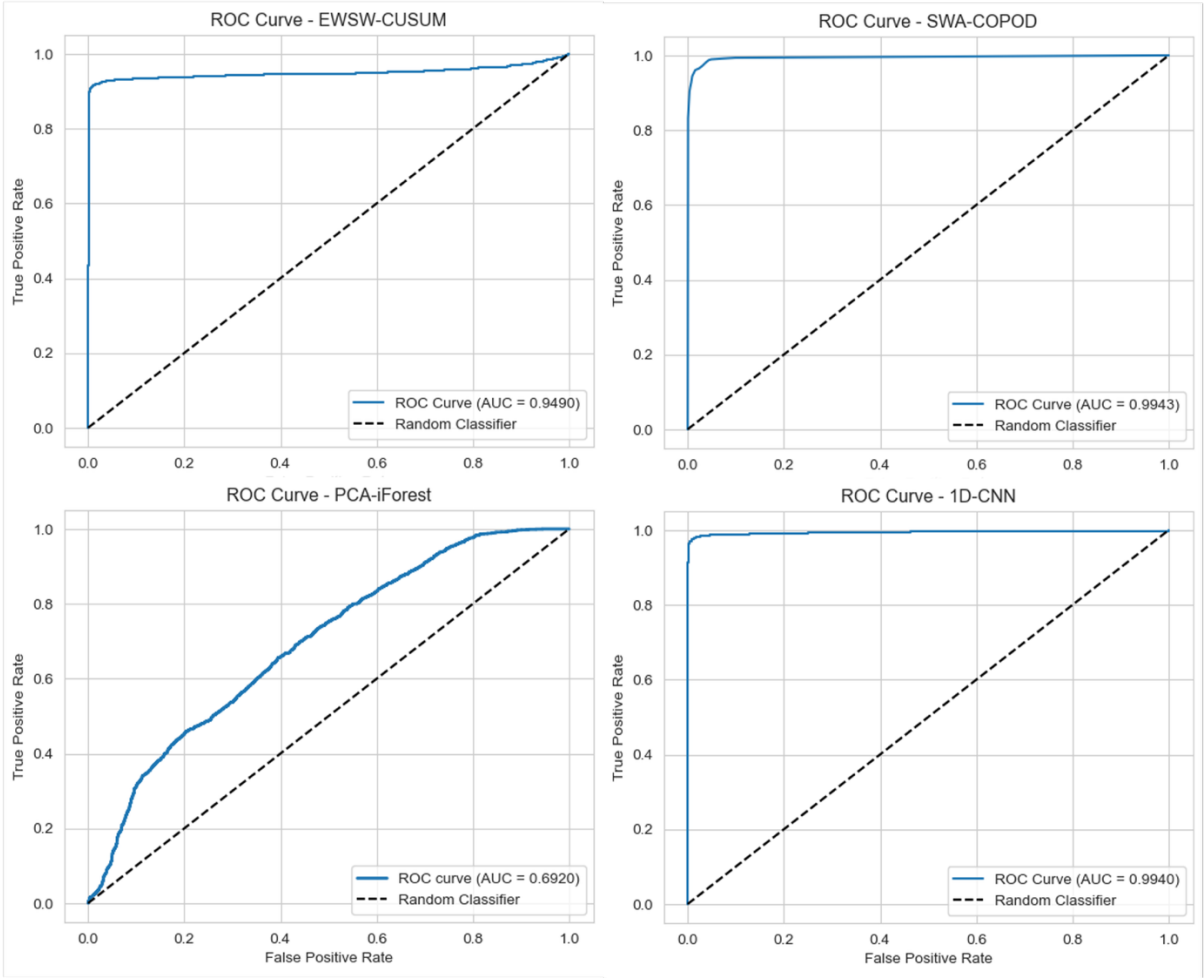

Figure 18. ROC curves with AUC values

As Figure 18 shows, SWA-COPOD achieved the highest AUC value of 0.9943, indicating excellent anomaly detection performance. It was closely followed by 1D-CNN with an AUC of 0.9940 and EWSW-CUSUM with 0.9490, both demonstrating

strong ranking capabilities. In contrast, PCA-iForest performed relatively poorly, with an AUC of only 0.6920, suggesting weaker ranking ability in this task. Overall, 1D-CNN and the two proposed methods, SWA-COPOD and EWSW-CUSUM, exhibit a clear performance advantage in this task.

Finally, the experimental results were also affected by the structure of the ECG5000 dataset. After splitting out the validation set, only 400 normal samples were left for training, while the test set contained 4500 samples. This setup, a small training set with a large test set, can limit the generalization of some models. However, under this challenging setting, the statistical models I proposed still performed reliably, further proving their practicality in small-sample scenarios.

5 Conclusion and Outlook

This study systematically examined four unsupervised anomaly detection algorithms. The first two, CUSUM and COPOD, are statistical methods; iForest represents an ensemble learning approach; and 1D-CNN is a deep learning method. For the first three, I proposed adaptations to better handle non-stationary time series: EWSW-CUSUM, SWA-COPOD, and PCA-iForest. Using ECG5000 as a unified benchmark dataset, I conducted modeling and testing, and the results show that EWSW-CUSUM and SWA-COPOD achieved anomaly detection performance comparable to the neural network-based 1D-CNN.

The initial goal of this study was to explore potential improvements to existing methods so they could perform well on non-stationary sequences. EWSW-CUSUM incorporates a sliding window within the sequence, exponential weighting, and a consecutive alarm mechanism. SWA-COPOD introduces multi-sample sliding windows and prediction aggregation. These mechanisms are entirely based on statistics and rules, without assuming any underlying data distribution, and therefore have strong transferability. Although PCA-iForest did not show significant improvement, PCA as a dimensionality reduction technique still offers valuable reference for anomaly detection in high-dimensional datasets. Overall, I believe this work successfully achieved its intended objectives.

For future work, the first step is to evaluate these algorithms on a wider range of datasets and re-assess their applicability and limitations. Another direction is to apply the proposed optimization mechanisms to other algorithms beyond the scope of this study to test whether better performance can be achieved. Given the complexity of anomaly detection scenarios, exploring broader usage of these methods is also worth considering—for example, adapting them for supervised or semi-supervised anomaly detection tasks, or integrating them into ensemble learning frameworks to overcome the limitations of any single method.

6 References

1. Wilson, Granville Tunnicliffe. “Time Series Analysis: Forecasting and Control, 5th Edition, by George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel and Greta M. Ljung, 2015. Published by John Wiley and Sons Inc., Hoboken, New Jersey, Pp. 712. ISBN: 978-1-118-67502-1.” Journal of Time Series Analysis 37, no. 5 (September 2016): 709–11. https://doi.org/10.1111/jtsa.12194.
2. Tsay, Ruey S. Multivariate Time Series Analysis: With R and Financial Applications. Wiley Series in Probability and Statistics. Hoboken, New Jersey: John Wiley & Sons, 2014.
3. Shumway, Robert H., and David S. Stoffer. Time Series Analysis and Its Applications: With R Examples. Springer Texts in Statistics. Cham: Springer Nature Switzerland, 2025. https://doi.org/10.1007/978-3-031-70584-7
4. Chen, Jie, and Arjun K. Gupta. *Parametric Statistical Change Point Analysis: With Applications to Genetics, Medicine, and Finance*. Boston: Birkhäuser Boston, 2012. https://doi.org/10.1007/978-0-8176-4801-5.
5. Lehman, Li-wei H., Ryan P. Adams, Louis Mayaud, George B. Moody, Atul Malhotra, Roger G. Mark, and Shamim Nemati. “A Physiological Time Series Dynamics-Based Approach to Patient Monitoring and Outcome Prediction.” *IEEE Journal of Biomedical and Health Informatics* 19, no. 3 (May 2015): 1068–76. https://doi.org/10.1109/JBHI.2014.2330827.
6. Guerard, John, Dimitrios Thomakos, and Foteini Kyriazi. “Automatic Time Series Modeling and Forecasting: A Replication Case Study of Forecasting Real GDP, the Unemployment Rate and the Impact of Leading Economic Indicators.” Edited by Xibin Zhang. *Cogent Economics & Finance* 8, no. 1 (January 1, 2020): 1759483. https://doi.org/10.1080/23322039.2020.1759483.
7. Sezer, Omer Berat, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. “Financial Time Series Forecasting with Deep Learning : A Systematic Literature Review: 2005–2019.” *Applied Soft Computing* 90 (May 2020): 106181. https://doi.org/10.1016/j.asoc.2020.106181.
8. Kumar, Raghavendra, Pardeep Kumar, and Yugal Kumar. “Time Series Data Prediction Using IoT and Machine Learning Technique.” *Procedia Computer Science* 167 (2020): 373–81. https://doi.org/10.1016/j.procs.2020.03.240.

9. Chandola, Varun, Arindam Banerjee, and Vipin Kumar. “Anomaly Detection: A Survey.” *ACM Computing Surveys* 41, no. 3 (July 2009): 1–58. https://doi.org/10.1145/1541880.1541882.
10. Wang, Hongzhi, Mohamed Jaward Bah, and Mohamed Hammad. “Progress in Outlier Detection Techniques: A Survey.” *IEEE Access* 7 (2019): 107964–0. https://doi.org/10.1109/ACCESS.2019.2932769.
11. Thudumu, Srikanth, Philip Branch, Jiong Jin, and Jugdutt Singh. “A Comprehensive Survey of Anomaly Detection Techniques for High Dimensional Big Data.” *Journal of Big Data* 7, no. 1 (December 2020): 42. https://doi.org/10.1186/s40537-020-00320-x.
12. Page, E. S. “CONTINUOUS INSPECTION SCHEMES.” Biometrika 41, no. 1–2 (1954): 100–115. https://doi.org/10.1093/biomet/41.1-2.100.
13. Lorden, G. “Procedures for Reacting to a Change in Distribution.” *The Annals of Mathematical Statistics* 42, no. 6 (December 1971): 1897–1908. https://doi.org/10.1214/aoms/1177693055.
14. Moustakides, George V. “Optimal Stopping Times for Detecting Changes in Distributions.” *The Annals of Statistics* 14, no. 4 (December 1, 1986). https://doi.org/10.1214/aos/1176350164.
15. Olufowobi, Habeeb, Uchenna Ezeobi, Eric Muhati, Gaylon Robinson, Clinton Young, Joseph Zambreno, and Gedare Bloom. “Anomaly Detection Approach Using Adaptive Cumulative Sum Algorithm for Controller Area Network.” In *Proceedings of the ACM Workshop on Automotive Cybersecurity*, 25–30. Richardson Texas USA: ACM, 2019. https://doi.org/10.1145/3309171.3309178.
16. Gualandi, Gabriele, and Alessandro V. Papadopoulos. “Worst-Case Impact Assessment of Multi-Alarm Stealth Attacks Against Control Systems with CUSUM-Based Anomaly Detection.” In *2023 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, 117–26. Toronto, ON, Canada: IEEE, 2023. https://doi.org/10.1109/ACSOS58161.2023.00029.
17. Qu, Jiahui, Qian Du, Yunsong Li, Long Tian, and Haoming Xia. “Anomaly Detection in Hyperspectral Imagery Based on Gaussian Mixture Model.” *IEEE Transactions on Geoscience and Remote Sensing* 59, no. 11 (November 2021): 9504–17. https://doi.org/10.1109/TGRS.2020.3038722.

18. Goldstein, M., & Dengel, A. (2012). *Histogram-based Outlier Score (HBOS): A fast Unsupervised Anomaly Detection Algorithm*. https://www.goldiges.de/publications/HBOS-KI-2012.pdf
19. Schölkopf, Bernhard, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. "Estimating the Support of a High-Dimensional Distribution." *Neural Computation* 13, no. 7 (July 1, 2001): 1443–71. https://doi.org/10.1162/089976601750264965.
20. Huang, Xunhua, Fengbin Zhang, Haoyi Fan, Huihui Chang, Bing Zhou, and Zuoyong Li. "Pseudo Anomalies Enhanced Deep Support Vector Data Description for Electrocardiogram Quality Assessment." *Computers in Biology and Medicine* 170 (March 2024): 107928. https://doi.org/10.1016/j.compbiomed.2024.107928.
21. Peng, Zhimin, Prudhvi Gurram, Heesung Kwon, and Wotao Yin. "Sparse Kernel Learning-Based Feature Selection for Anomaly Detection." *IEEE Transactions on Aerospace and Electronic Systems* 51, no. 3 (July 2015): 1698–1716. https://doi.org/10.1109/TAES.2015.130730.
22. Navarro-Acosta, Jesús Alejandro, Irma D. García-Calvillo, Vanesa Avalos-Gaytán, and Edgar O. Reséndiz-Flores. "Metaheuristics and Support Vector Data Description for Fault Detection in Industrial Processes." *Applied Sciences* 10, no. 24 (December 21, 2020): 9145. https://doi.org/10.3390/app10249145.
23. Said Elsayed, Mahmoud, Nhien-An Le-Khac, Soumyabrata Dev, and Anca Delia Jurcut. "Network Anomaly Detection Using LSTM Based Autoencoder." In *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, 37–45. Alicante Spain: ACM, 2020. https://doi.org/10.1145/3416013.3426457.
24. Shrestha, Rakesh, Mohammadreza Mohammadi, Sima Sinaei, Alberto Salcines, David Pampliega, Raul Clemente, Ana Lourdes Sanz, Ehsan Nowroozi, and Anders Lindgren. "Anomaly Detection Based on LSTM and Autoencoders Using Federated Learning in Smart Electric Grid." *Journal of Parallel and Distributed Computing* 193 (November 2024): 104951. https://doi.org/10.1016/j.jpdc.2024.104951.
25. Kiranyaz, Serkan, Morteza Zabihi, Ali Bahrami Rad, Turker Ince, Ridha Hamila, and Moncef Gabbouj. "Real-Time Phonocardiogram Anomaly Detection by Adaptive 1D Convolutional Neural Networks." *Neurocomputing* 411 (October 2020): 291–301. https://doi.org/10.1016/j.neucom.2020.05.063.

26. Chen, Yanping, Yuan Hao, Thanawin Rakthanmanon, Jesin Zakaria, Bing Hu, and Eamonn Keogh. “A General Framework for Never-Ending Learning from Time Series Streams.” Data Mining and Knowledge Discovery 29, no. 6 (November 2015): 1622–64. https://doi.org/10.1007/s10618-014-0388-4.
27. Dau, Hoang Anh, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. “The UCR Time Series Archive.” arXiv, 2018. https://doi.org/10.48550/ARXIV.1810.07758.
28. Ahmed, Ejaz, Andrew Clark, and George Mohay. “A Novel Sliding Window Based Change Detection Algorithm for Asymmetric Traffic.” In *2008 IFIP International Conference on Network and Parallel Computing*, 168–75. Shanghai, China: IEEE, 2008. https://doi.org/10.1109/NPC.2008.81.
29. Montes De Oca, Veronica, Daniel R. Jeske, Qi Zhang, Carlos Rendon, and Mazda Marvasti. “A Cusum Change-Point Detection Algorithm for Non-Stationary Sequences with Application to Data Network Surveillance.” *Journal of Systems and Software* 83, no. 7 (July 2010): 1288–97. https://doi.org/10.1016/j.jss.2010.02.006.
30. Li, Zheng, Yue Zhao, Nicola Botta, Cezar Ionescu, and Xiyang Hu. “COPOD: Copula-Based Outlier Detection.” In 2020 IEEE International Conference on Data Mining (ICDM), 1118–23. Sorrento, Italy: IEEE, 2020. https://doi.org/10.1109/ICDM50108.2020.00135.
31. A. Sklar, “Fonctions de repartition an dimensions et leurs marges,” *Publ. inst. statist. univ. Paris*, vol. 8, pp. 229–231, 1959.
32. Mann, Douglas L., Douglas P. Zipes, Peter Libby, Robert O. Bonow, and Eugene Braunwald, eds. Braunwald’s Heart Disease: A Textbook of Cardiovascular Medicine. Tenth edition. Philadelphia, PA: Elsevier/Saunders, 2015.
33. Haïssaguerre, Michel, Pierre Jaïs, Dipen C. Shah, Atsushi Takahashi, Mélèze Hocini, Gilles Quiniou, Stéphane Garrigue, Alain Le Mouroux, Philippe Le Métayer, and Jacques Clémenty. “Spontaneous Initiation of Atrial Fibrillation by Ectopic Beats Originating in the Pulmonary Veins.” New England Journal of Medicine 339, no. 10 (September 3, 1998): 659–66. https://doi.org/10.1056/NEJM199809033391003.
34. Issa, Ziad F., John M. Miller, and Douglas P. Zipes. “Electrophysiological Mechanisms of Cardiac Arrhythmias.” In Clinical Arrhythmology and

Electrophysiology: A Companion to Braunwald's Heart Disease, 36–61. Elsevier, 2012. https://doi.org/10.1016/B978-1-4557-1274-8.00003-8.

35. Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation Forest." In *2008 Eighth IEEE International Conference on Data Mining*, 413–22. Pisa, Italy: IEEE, 2008. https://doi.org/10.1109/ICDM.2008.17.
36. Knuth, Donald Ervin. *The Art of Computer Programming*. Second ed. Reading (Mass.) London Manila [etc.]: Addison-Wesley publ, 1973.
37. Lecun, Y., L. Bottou, Y. Bengio, and P. Haffner. "Gradient-Based Learning Applied to Document Recognition." *Proceedings of the IEEE* 86, no. 11 (November 1998): 2278–2324. https://doi.org/10.1109/5.726791.
38. Abdel-Hamid, Ossama, Abdel-rahman Mohamed, Hui Jiang, and Gerald Penn. "Applying Convolutional Neural Networks Concepts to Hybrid NN-HMM Model for Speech Recognition." In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4277–80. Kyoto, Japan: IEEE, 2012. https://doi.org/10.1109/ICASSP.2012.6288864.
39. Zhang, Yong, Meng Joo Er, Rui Zhao, and Mahardhika Pratama. "Multiview Convolutional Neural Networks for Multidocument Extractive Summarization." *IEEE Transactions on Cybernetics* 47, no. 10 (October 2017): 3230–42. https://doi.org/10.1109/TCYB.2016.2628402.
40. Kiranyaz, Serkan, Morteza Zabihi, Ali Bahrami Rad, Turker Ince, Ridha Hamila, and Moncef Gabbouj. "Real-Time Phonocardiogram Anomaly Detection by Adaptive 1D Convolutional Neural Networks." *Neurocomputing* 411 (October 2020): 291–301. https://doi.org/10.1016/j.neucom.2020.05.063.
41. Kiranyaz, Serkan, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J. Inman. "1D Convolutional Neural Networks and Applications: A Survey." *Mechanical Systems and Signal Processing* 151 (April 2021): 107398. https://doi.org/10.1016/j.ymssp.2020.107398.
42. Hamed Mozaffari, M., and Li-Lin Tay. "Overfitting One-Dimensional Convolutional Neural Networks for Raman Spectra Identification." *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy* 272 (May 2022): 120961. https://doi.org/10.1016/j.saa.2022.120961.
43. Roberts, S. W. "Control Chart Tests Based on Geometric Moving Averages." *Technometrics* 1, no. 3 (August 1959): 239–50. https://doi.org/10.1080/00401706.1959.10489860.

44. Crowder, Stephen V. “Design of Exponentially Weighted Moving Average Schemes.” *Journal of Quality Technology* 21, no. 3 (July 1989): 155–62. https://doi.org/10.1080/00224065.1989.11979164.
45. De Vargas, Vera Do Carmo C., Luis Felipe Dias Lopes, and Adriano Mendonça Souza. “Comparative Study of the Performance of the CuSum and EWMA Control Charts.” *Computers & Industrial Engineering* 46, no. 4 (July 2004): 707–24. https://doi.org/10.1016/j.cie.2004.05.025.