FENet: Functional Embedding Neural Network for Change-Point Detection in Functional Time Series

Caixia Xu[1], Jinhong You[1], Yiming Tang[2,4]*, Wen Li[3], Shouguo Du[4], Jiguo Cao[5]

[1] Shanghai University of Finance and Economics
[2] Shanghai Lixin University of Accounting and Finance
[3] Shanghai University of International Business and Economics
[4]Shanghai Municipal Big Data Center
[5] Simon Fraser University

xucaixia@163.sufe.edu.cn, johnyou07@163.com, tang.yiming@lixin.edu.cn
liwen@suibe.edu.cn, shouguo.du.sh@outlook.com, jiguo_cao@sfu.ca

Abstract

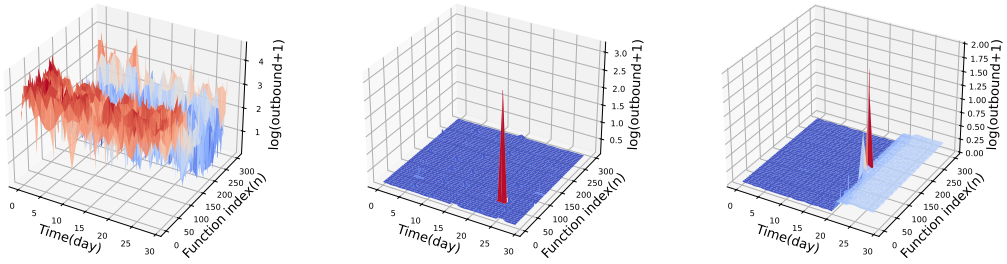
Detecting abnormal changes in server operational data is an important task in modern computing systems, as unexpected shifts in traffic, workload, or performance metrics may indicate service degradation, abnormal access patterns, system failures, or potential security risks. However, server monitoring data are often naturally observed as functional time series, with complex temporal patterns, high dimensionality, and heterogeneous structures, making traditional change-point detection methods difficult to apply reliably in practice. To address this problem, we propose a general change-point detection framework based on a Functional Embedding Neural Network (FENet), designed to provide robust detection across different data types and structural conditions without relying on strong prior assumptions. The proposed method is motivated by the connection between the classical functional CUSUM (FCUSUM) statistic and neural network representations, which allows FENet to learn effective detection rules directly from data while retaining useful statistical intuition. We provide theoretical analysis to formalize this connection and study key properties of the proposed approach. Extensive simulation studies show that FENet achieves competitive performance compared with existing methods and remains robust under limited training data and distributional mismatch between training and test settings. We further apply FENet to real-world server operational metrics, demonstrating its practical value in detecting and localizing abnormal changes in complex server monitoring systems.

1 Introduction

Our study is motivated by the practical problem of detecting structural changes in the operational data of computer servers. In modern computing systems, servers continuously generate high-frequency monitoring data, such as inbound and outbound traffic, workload, and other performance-related metrics. Careful analysis of such data is important for maintaining system security, operational sustainability, and economic efficiency, since abnormal changes may indicate hardware failures, unusual access patterns, service degradation, or potential data leakage. In our application, server traffic is recorded at five-minute intervals, so each day naturally forms an intraday traffic profile. As illustrated in Figure 1, these profiles may exhibit regular but complex patterns and can contain local spikes or short-term fluctuations even under normal operating conditions. Such isolated peaks, as in Figure 1b, are not necessarily meaningful structural changes, but may be caused by recording errors, transient pulses, or short-lived bursts of requests. Our interest is instead in persistent changes across

*Corresponding author

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



(a) Regular but complex pattern (b) Isolated spike (c) Persistent structural change

Figure 1: Outbound traffic dynamics of different servers over a 30-day period. Each surface represents daily intraday traffic profiles over calendar days.

days, where the entire daily traffic profile shifts in level, shape, or variability after a certain time point, as shown in Figure 1c. Detecting this type of day-level structural change helps system administrators identify which server has changed and when the change begins, thereby supporting timely diagnosis and response.

From a statistical perspective, this problem is naturally related to change-point detection in functional time series. Because each day consists of a complete intraday trajectory rather than a single scalar observation, the monitoring sequence is more appropriately viewed as a sequence of functional observations. Functional time series data have proved useful in many areas, and a diverse range of functional time series models have been introduced and studied over the last twenty years (e.g. [Ramsay and Silverman, 2005, Ferraty and Vieu, 2006, Hörmann and Kokoszka, 2010, Horváth and Kokoszka, 2012]).Due to the frequent violation of the stationarity assumption in both traditional and functional time series, detecting structural breaks has become an important task. For instance, Horváth et al. [2014] identified change points in intraday price curves of certain U.S. stocks, while Li et al. [2023] provided evidence of nonstationary behavior in functional time series constructed from age- and sex-specific life-table death counts. In the server monitoring problem considered here, directly applying conventional time series change-point methods to the raw five-minute observations may be misleading, since the pronounced intraday pattern can dominate the signal of interest. A more appropriate strategy is to treat each daily traffic curve as one functional observation and then detect persistent changes across days.

There has been a growing focus on detecting structural breaks in functional time series. Broadly, two types of detection techniques are prevalent. The first involves reducing the infinite dimension of functional data to a finite dimension by functional principal component analysis (e.g. [Aue et al., 2009, Berkes et al., 2009, Zhang et al., 2011, Aston and Kirch, 2012, Aue et al., 2014, Torgovitski, 2015, Gromenko et al., 2017, Aue and Van Delft, 2020]). The second approach is a fully functional detection method that bypasses preliminary dimension reduction (e.g. [Horváth et al., 2014, Sharipov et al., 2016, Aue et al., 2018, Li et al., 2024a]) via functional CUSUM (FCUSUM) statistics. These CUSUM-based detection methods possess the capability to identify breaks that may be orthogonal to the projection subspace determined by the dimension reduction process. However, as unsupervised procedures, these methods do not leverage labeled data to adaptively learn informative structures and instead rely on statistical thresholds, which may limit their efficiency and detection power. These limitations become particularly relevant in server monitoring, where labeled historical cases may be available and the patterns of interest can differ substantially across servers.

This paper focuses on developing a more robust and efficient functional change-point detection method that can adapt to diverse data characteristics. Neural networks, as data-driven supervised learning methods, provide a natural tool for this purpose, since they can learn informative structures from labeled examples rather than relying entirely on pre-specified statistics or fixed thresholds. In recent years, neural-network-based change-point detection has been explored in several directions. Chang et al. [2019] proposed the KL-CPD method to address the challenges of kernel selection and limited sample sizes in kernel two-sample change-point tests. Liehrmann et al. [2021] employed neural networks to detect multiple change points in over-dispersed data. Gupta et al. [2022] introduced a three-stage adaptive framework for detecting complex change points in multivariate time series. Lattari et al. [2022] developed a neural architecture based on long short-term memory (LSTM) cells to

capture temporal dependence within time series data. Lee et al. [2023] designed a neural network that computes the CUSUM statistic sequentially for online change-point detection. Particularly relevant to our work, Li et al. [2024b] introduced a neural architecture for several types of change-point detection problems and showed that CUSUM-type statistics can be represented within a neural network framework. This connection provides a useful bridge between classical change-point statistics and supervised learning. However, most existing neural approaches are designed for scalar, multivariate, or online time series settings, and their connection to functional change-point detection remains limited. This motivates us to develop a neural network framework tailored to functional time series, while retaining the statistical intuition of FCUSUM-type procedures.

Motivated by this connection, we develop a neural architecture tailored to functional time series by embedding the FCUSUM statistic into a trainable network structure. Following this guiding principle, our main contributions are outlined as follows.

1. We propose FENet, a neural architecture for functional change-point detection that recovers the classical FCUSUM statistic as a special case, while extending it to a data-adaptive framework. To the best of our knowledge, there has been no prior work utilizing neural networks for supervised change-point detection in functional time series.

2. We establish theoretical guarantees that characterize the expressivity and statistical properties of FENet, showing that it can faithfully represent FCUSUM while offering a richer function class capable of capturing more complex change-point structures.

3. We demonstrate through simulations and real-world experiments that FENet achieves more robust and balanced performance under distributional mismatch and complex data settings.

The paper is organized as follows. Section 2 introduces the proposed methodology for detecting change points in functional time series. Section 3 presents the theoretical results, showing the relationship between FCUSUM and the FENet, as well as the effectiveness of FCUSUM and the detection accuracy. In Section 4, numerical studies are conducted to evaluate the performance of our method. Section 5 illustrates the application of our approach to identifying change points in a server operational metrics dataset, highlighting its practical relevance. Finally, Section 6 offers concluding remarks, with supplementary notations, proofs, implementation details, and extra experiments provided in the Appendix. The codes needed to replicate the experiments can be found at https://github.com/Yiming-Tang/FENet.

2 Methodology

We begin by introducing notations. In this paper, let $\mathcal{H}$ be the Hilbert space defined as a set of measurable functions $g(\cdot)$ on a bounded set $\mathbb{C}$ such that $\int_{\mathbb{C}} g^2(s)ds < \infty$. The inner product in $\mathcal{H}$ is defined as $\langle g_1(s), g_2(s)\rangle = \int_{\mathbb{C}} g_1(s)g_2(s)ds$. For $g \in \mathcal{H}$, define the canonical norm $||g|| = \langle g, g\rangle^{1/2} = [\int_{\mathbb{C}} |g(s)|^2 ds]^{1/2}$, $||g||_p := [\int_{\mathbb{C}} |g(s)|^p ds]^{1/p}$. Let $|A|$ be the number of elements in set A. Let $\mathbf{1}_a$ be the all-one vector of length a. For any $n \in \mathbb{Z}^+$, we define $[n] := 1, \ldots, n$. For a vector $\mathbf{a} = (a_1, \cdots, a_p)^T$, we define $||\mathbf{a}||_2 = \sqrt{\mathbf{a}^T\mathbf{a}} = (\sum_{i=1}^p a_i^2)^{1/2}$ as its L_2-norm.

2.1 Functional Time Series with Change Points

Let $\mathbf{X}(s) = \{X_t(s), s \in [0,1]\}, t = 1, \ldots, T$ be a sequence of functional time series, where each $X_t(s)$ is defined on $L_2([0,1])$, a Hilbert space of square-integrable functions with the inner product $\langle f, g\rangle = \int_0^1 f(s)g(s)ds$ for any $f, g \in L_2([0,1])$. We assume that the mean function of $\{X_t(s)\}_{t=1}^T$ exists m structural breaks at the positions $\{t_k^*\}_{k=1}^m$, where the total number of change-points m is unknown. When $m \geq 1$, let $\mathcal{M} = \{t_1^*, \cdots, t_m^*\}$ denote the set of the m change-points satisfying $1 \equiv t_0^* < t_1^* < \cdots < t_m^* < t_{m+1}^* \equiv T$, by which the whole time span is divided into $m+1$ regimes, denoted by $I_k = (t_k^*, t_{k+1}^*]$ for $k = 0, 1, \cdots, m$. Furthermore, let $\mu_k(s)$ be the mean function of the $X_t(s)$'s with the associated time t lying in the segment I_k, where $\{\mu_k(s)\}_{k=0}^m$ is the set of functions. Assume $\mu_k(s) \neq \mu_{k+1}(s)$ for model identifiability.

Under these settings, the multiple change-points model for the functional time series $\{X_t(s)\}_{t=1}^T$ is

$$X_t(s) = \sum_{k=0}^{m} \mu_k(s)\mathbb{I}_{(t_k^*, t_{k+1}^*]}(t) + \epsilon_t(s), \tag{1}$$

where $\mathbb{I}_{(t_k^*, t_{k+1}^*]}(t) = 1$ if $t \in (t_k^*, t_{k+1}^*]$ and 0 otherwise, and $\epsilon_t(s)$ is a functional time series. The assumptions for the innovations $\epsilon_t(s)$ are stated in Section 3. Our interest lies in detecting the number of change points m and their positions $t_1^*, \cdots, t_m^*$ in the functional time series $\{X_t(s)\}_{t=1}^T$.

2.2 Functional CUSUM Statistic

In this section, we employ a statistic based on the CUSUM for detecting the breaks of the functional time series with one change-point, that is $m = 1$, and the functional model (1) can be switched to

$$X_t(s) = \mu(s) + \delta(s)\mathbb{I}(t > t^*) + \epsilon_t(s), \tag{2}$$

where $\mu(s)$ is the baseline mean function and $\delta(s)$ is the mean shift function after the break time t^*. We define the FCUSUM statistic based on Aue et al. [2018] as:

$$U_t(s) = \frac{1}{\sqrt{T}}(\sum_{j=1}^{t} X_j(s) - \frac{t}{T}\sum_{j=1}^{T} X_j(s)),$$

where $1 \leq t < T$, by which we obtain a set of functional time series $\{U_t(s)\}_{t=1}^{T-1}$, which is difficult to identify its maximum due to the infinite-dimensional nature of functional data.

Therefore, we need to make further transformations to the FCUSUM series $U_t(s)$. The canonical norm of a function allows $U_t(s)$ to be converted into a standard time series. This conversion is referred to as the transformed FCUSUM statistic, which is defined by

$$M_t = ||U_t(s)||^2. \tag{3}$$

When a change point exists, M_t attains a maximum at the change point location. Let $\mathbf{M} := (M_1, \cdots, M_{T-1})^\top$. The FCUSUM test for detecting a change in the mean function is defined as

$$h_\lambda(\mathbf{X}(s)) = \mathbb{I}\{||\mathbf{M}||_\infty > \lambda\}. \tag{4}$$

The position of the change point can be detected by $\hat{t} = \arg\max_{1 \leq t < T} M_t$.

In Section 2.3, we present specific methodologies, where the transformed FCUSUM statistic series $\mathbf{M}$ is reformulated within a neural network framework. The theory that FCUSUM test can be represented as a neural network is a crucial basis for the method, which will be demonstrated in Theorem 1.

2.3 Functional Embedding Neural Network (FENet)

In this section, we demonstrate how the task of identifying the existence of change points, as formulated in Equation (4), can be reframed as a binary classification problem and represented using a functional embedding neural network. The structure of the FENet is illustrated in Figure 2. While the main development of our method is based on a multilayer perceptron (MLP) architecture, we further explore the use of convolutional neural networks (CNNs) as an extension in the empirical study.

We first introduce Lemma 1, which describes the functional embedding of the functional time series data $\mathbf{X}(s)$, mapping the original infinite-dimensional functions into a finite-dimensional representation suitable for network input.

Lemma 1 *Let $\tilde{\mathbf{x}}_t = \mathbf{S}_\mathbf{X}\mathbf{v}_t$ be a vector of T dimensions, and $\tilde{\mathbf{x}} = (\tilde{\mathbf{x}}_1^\top, \cdots, \tilde{\mathbf{x}}_{T-1}^\top)^\top$ be a vector of $T(T-1)$ dimensions, where $\mathbf{v}_t = ((\frac{1}{T} - \frac{t}{T^2}) \cdot \mathbf{1}_t, -\frac{t}{T^2} \cdot \mathbf{1}_{T-t})^\top$ for $t = 1, \cdots, T-1$, and*

$$\mathbf{S}_\mathbf{X} := \langle \mathbf{X}(s), \mathbf{X}(s) \rangle = \begin{pmatrix} V_{11} & V_{12} & \cdots & V_{1T} \\ V_{21} & V_{22} & \cdots & V_{2T} \\ \cdots & \cdots & \cdots & \cdots \\ V_{T1} & V_{T2} & \cdots & V_{TT} \end{pmatrix}_{T \times T},$$

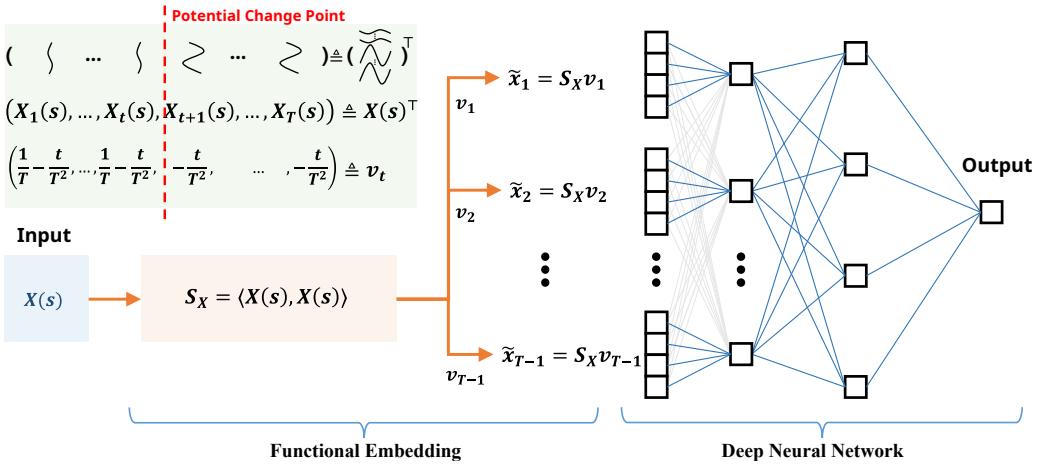


Figure 2: The overall architecture of our FENet. We first embed $\mathbf{X}(\mathbf{s})$ into a sequence of representations $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_{T-1}$, which are then fed into the deep neural network for change-point detection.

where V_{ij} is defined by $= \langle X_i(s), X_j(s) \rangle$. Let $U(\tilde{\mathbf{x}}) := (\mathbf{v}_1'^\top \tilde{\mathbf{x}}_1, \cdots, \mathbf{v}_{T-1}'^\top \tilde{\mathbf{x}}_{T-1})^\top$. Then the FCUSUM test, defined in Equation (4)*, for a change in the mean function of the data is equivalent to*

$$h_\lambda(\mathbf{X}(s)) = \mathbb{I}\{||U(\tilde{\mathbf{x}})||_\infty > \lambda\}. \tag{5}$$

Following Lemma 1, we express the FCUSUM statistic in Equation (4) as a function of $\tilde{\mathbf{x}}$, the output of the functional embedding layer, which serves as the input to the deep neural network.

Next, we construct a neural network class $\mathcal{H}_{L,\mathbf{n}}$ with an input layer of dimension $T(T-1)$ and a single output node, where $L \in \mathbb{Z}^+$ denotes the number of hidden layers, and $\mathbf{n} = (n_1, \ldots, n_L)^\top$ represents the vector of hidden layer widths, with n_i specifying the number of nodes in the ith hidden layer. By convention, $n_0 = T(T-1)$ and $n_{L+1} = 1$. We further define the shifted activation function $\sigma_{\mathbf{b}} : \mathbb{R}^r \to \mathbb{R}^r$ as $\sigma_{\mathbf{b}}((y_1, \cdots, y_r)^\top) = (\sigma(y_1 + b_1), \cdots, \sigma(y_r + b_r))^\top$ for any bias $\mathbf{b} = (b_1, \cdots, b_r)^\top \in \mathbb{R}^r$, where $\sigma(x) = \max(x, 0)$ represents the ReLU activation function. The neural network can be formally expressed as the composite function $h : \mathbb{R}^{T(T-1)} \to \{0, 1\}$ as

$$h(\tilde{\mathbf{x}}) := \sigma_\lambda^* W_L \sigma_{\mathbf{b}_L} W_{L-1} \sigma_{\mathbf{b}_{L-1}} \cdots W_1 \sigma_{\mathbf{b}_1} W_0 \tilde{\mathbf{x}}, \tag{6}$$

where $\sigma_\lambda^*(x) = \mathbb{I}\{x > \lambda\}$, $W_l \in \mathbb{R}^{n_{l+1} \times n_l}$ for $l \in \{0, \cdots, L\}$ represent the weight matrices.

We use the 0-1 loss, which emphasizes the percentage of samples correctly classified, as a fundamental metric for binary classification task. The output layer in Equation (6) utilizes the shifted heaviside function $\sigma_\lambda^*(x)$, which chose the binary classification as the final activation function. In addition to its widespread application in machine learning, another benefit of the 0-1 loss is its compatibility with the theory of the Vapnik-Chervonenkis (VC) dimension (Sec 6, Shalev-Shwartz and Ben-David [2014]), allowing for the bounding of the generalization error of a binary classifier.

2.4 Multiple Change-Points Detection

Additionally to testing for changes, the FENet classifier can be used to estimate the number and locations of multiple change points. Here, we propose a methodology for deriving a change-point location estimator from a classifier, in the spirit of the MOSUM approach Eichinger and Kirch [2018]. The approach involves iteratively employing a classifier ψ on data extracted from a sliding window of size T. The duration of the individual samples in the dataset is denoted as T^*.

We partition all data into sub-samples $X^*_{[i,T+i)}(s) = (X_i(s), \cdots, X_{T+i-1}(s))$, $i = 1, \cdots, T^* - T + 1$. Following the steps outlined in Section 2.2, we transform these sub-samples into vectors of length $T(T-1)$, denoted as $\tilde{\mathbf{x}}_i$, which are then fed into the pre-trained FENet classifier ψ. When the sub-sample $X^*_{[i,T+i)}(s)$ contains a change point, the output of $\psi(\tilde{\mathbf{x}}_i)$ will be 1. Otherwise, the

classifier will output 0. Therefore, by thresholding a moving average of the classifier outputs over the sliding samples, we can identify the positions of the change points. Here the thresholding parameter $\gamma \in \mathbb{R}^+$ is chosen to be $1/2$. The general algorithm is outlined in Appendix B.1.

3 Asymptotic Properties

With the proposed FENet and algorithms in place, we next examine their theoretical properties. Throughout the analysis, we assume that the innovations $(\epsilon_t(s) : t \in \mathbb{Z})$ in model (1) are weakly dependent and constitute a stationary functional time series, as specified in Assumption 1.

Assumption 1 *The innovations $(\epsilon_t(s) : t \in \mathbb{Z})$ satisfy $[i]$ there exists a measurable function $f : S^\infty \to L^2[0,1]$, where S is an arbitrary measurable space, and independent and identically distributed (IID) innovations $(\eta_t(s) : t \in \mathbb{Z})$ with values in S, such that $\epsilon_t(s) = f(\eta_{t-1}(s), \cdots)$ for $t \in \mathbb{Z}$. $[ii]$ there are l-dependent sequences $(\epsilon_t^l(s) : t \in \mathbb{Z})$ such that, for some $p > 2$, $\sum_{l=1}^\infty (\mathbb{E}[||\epsilon_t(s) - \epsilon_t^l(s)||_p^p])^{1/p} < \infty$, where $\epsilon_t^l(s) = f(\eta_t(s), \cdots, \eta_{t-l+1}(s), \eta_{t-l}^*(s), \eta_{t-l-1}^*(s), \cdots)$ with $\eta_t^*(s)$ being independent copies of $\eta_t(s)$.*

Processes that adhere to Assumption 1 have been classified as L_p-l-approximable by Hörmann and Kokoszka [2010]. These processes encompass a wide range of stationary functional time series models, including functional autoregressive (FAR) as well as functional generalized autoregressive conditional heteroscedastic processes [Aue et al., 2017]. It is assumed that the underlying error innovations $(\epsilon_t(s) : t \in \mathbb{Z})$ are elements of an arbitrary measurable space S. However, in many practical settings, S is itself a function space, and thus $f(\eta_t(s), \eta_{t-1}(s), \cdots)$ should be interpreted as a functional of the past trajectory $(\eta_{t'}(s) : t' \leq t)$.

Theorem 1 below demonstrates that $h_\lambda(\tilde{\mathbf{x}})$ as described in Equation (5) can be equivalently represented by the network structure introduced in Equation (6).

Theorem 1 *For any $\lambda > 0$, we have $h_\lambda(\tilde{\mathbf{x}}) \in \mathcal{H}_{1,T-1}$.*

The proof of Theorem 1 is provided in Appendix A.2 . Recognizing that the widely used FCUSUM statistic can be interpreted as a simple neural network suggests that, with sufficient training data, a neural architecture encompassing the FCUSUM-based classifier as a special case should perform at least as effectively as FCUSUM in distinguishing change-point from non–change-point signals. Additional theoretical results of the FCUSUM statistic are deferred to the Appendix A.1.

Theorem 2, derived from Corollary 1, Theorem 7 of Bartlett et al. [2019] and Corollary 3.4 of Mohri et al. [2012], establishes that the empirical risk minimizer within the neural network class $\mathcal{H}_{1,T-1}$ exhibits strong generalization capabilities across the spectrum of change-point scenarios outlined in Equation (1). Given a set of training data $(\mathbf{X}(s)^{(1)}, Y^{(1)}), \cdots, (\mathbf{X}(s)^{(N)}, Y^{(N)}))$ and any function $h : \mathbf{X}(s) \to \{0,1\}$, the empirical risk of h is defined as $L_N(h) := \frac{1}{N}\sum_{i=1}^N \mathbb{I}\{Y^{(i)} \neq h(\mathbf{X}(s)^{(i)})\}$.

Theorem 2 *Under Assumption 1, define $Y = \mathbb{I}\{m \neq 0\}$, and suppose that the training data $\mathcal{D} = \{(\mathbf{X}(s)^{(i)}, Y^{(i)})\}_{i=1}^N$ consist of independent copies of $(\mathbf{X}(s), Y)$. Let $h_{ERM} := \arg\min_{h \in \mathcal{H}_{1,T-1}} L_N(h)$ denote the empirical risk minimizer. Then there exists a universal constant $C > 0$, $K_0 \in \mathbb{N}$ and $T_0 \in \mathbb{N}$ such that, for any $K > K_0$, $T > T_0$, $\lambda = K^2$ and $\varphi \in (0,1)$, the following bound holds with probability $1 - \varphi$.*

$$\mathbb{P}(h_{ERM}(\tilde{\mathbf{x}}) \neq Y | \mathcal{D}) \leq 2e^{-2K/\lambda_1} + C\sqrt{\frac{T^3 \ln(T) \ln(N) + \ln(1/\varphi)}{N}}, \tag{7}$$

where λ_1 is the same as that in Corollary 1.

The mis-classification error in Equation (7) consists of two components. The first corresponds to the intrinsic error of the FCUSUM classifier, as characterized in Corollary 1. The second depends on the complexity of the neural network class, measured by its VC dimension. As shown in Theorem 2, when the training sample size N is sufficiently larger than $T^3 \ln T$, a well-trained neural network with a single hidden layer and $T(T-1)$ hidden nodes achieves performance comparable to the FCUSUM classifier. In practice, empirical results suggest that satisfactory performance can often be achieved with a substantially smaller number of training samples. Analogous representation and generalization

results for a depthwise CNN class are given in Theorems 5 and 6 of Appendix A.1, with proofs in Appendix A.2.

Remark 1 *In fact, when the number of nodes in each layer satisfies certain conditions in Li et al. [2024b], the conclusion of Theorem 2 can be further extended to multi-layer neural networks. Our simulation experiments in Appendix C.2 also confirm this result, showing that multi-layer networks can achieve comparable or even higher classification accuracy. Since the same workflow is applied, we do not elaborate on this part here.*

Algorithm 1 serves as a foundational framework for transforming a change-point classifier into a location estimator. The following Theorem 3 demonstrates that when Algorithm 1 is implemented alongside a FCUSUM classifier, it achieves an optimal convergence rate for the task of change-point localization. Here we consider the model described in Equation (1) with time length T^*. For a sequence of change points $1 = t_0^* < t_1^* < \cdots, < t_m^* < T^* = t_{m+1}^*$, we have $\mu_t(s) = \mu_k(s)$ for all $t \in (t_k^*, t_{k+1}^*]$, with T representing the time length as well as the size of the sliding window.

Theorem 3 *Under Assumption 1, consider $X_1(s), \cdots, X_{T^*}(s)$ following Equation (1), with $1 = t_0^* < t_1^* < \cdots < t_m^* < T^* = t_{m+1}^*$ where $t_k^* - t_{k-1}^* \geq 2T$ for all $k \in [m+1]$ and $\max_{k\in[m]} ||\mu_k(s) - \mu_{k-1}(s)|| \geq 2\sqrt{2}C$, define $h_{ERM}(\tilde{\mathbf{x}})$ as in Theorem 2. Let $\hat{t}_1, \cdots, \hat{t}_{\hat{m}}$ be the results of Algorithm 1 with input $X_1(s), \cdots, X_{T^*}(s)$, $\psi = h_{ERM}(\tilde{\mathbf{x}})$ and $\gamma = \lfloor T/2 \rfloor / T$. Then for any $\epsilon > 0$, choose $K_0(\epsilon) = \lambda_1/2 \ln(2/\epsilon)$ and take any $K > K_0$ with $\lambda = K^2$, we can establish that $\mathbb{P}(\mathcal{A}_T) \geq 1 - \epsilon$, where $\mathcal{A}_T = \{\hat{m} = m; |t_k - \hat{t}_k| \leq 2C^2\rho^{-2}\}$ with $\rho = \max_{k\in[m]} ||\mu_k(s) - \mu_{k-1}(s)||$, and λ_1 is the same as that in Corollary 1.*

The theorem presented above demonstrates that by utilizing the FCUSUM change-point classifier $\psi = h_{ERM}(\tilde{\mathbf{x}})$ in combination with Algorithm 1, it becomes possible to consistently estimate both the number and locations of multiple change points within the data stream.

4 Real Data Analysis

In this section, we conduct an empirical analysis on a real-world dataset to demonstrate the effectiveness of the proposed method. The data consists of server operational metrics from the Shanghai Big Data Center. We collected 100 days of data, from September to December 2024, for 100 servers. The analysis primarily focuses on network inbound and outbound traffic data, which are sampled at a frequency of once every five minutes. Ground-truth change-point labels were manually annotated.

Due to the inherent operational characteristics of the machines and potential biases in data collection, the collected time series may contain transient anomalies or spikes. These phenomena, while not true change points, are often misidentified as such by traditional time series detection methods. Our objective is to detect structural changes in the daily operational patterns of the servers, which may be caused by anomalous programs or access patterns. To this end, we treat the 288 observations recorded per day for each metric and each server as a single functional observation, forming a functional time series. We then apply the proposed method to detect change points in these functional data. In particular, we apply a logarithmic transformation to the network's inbound and outbound traffic data using the function $\log(\log(x + 1) + 1)$.

To train the single change point detector, we define a time window of 7 days. This rolling window is applied across the 100-day period, yielding 94 segments for each server. The segments from all servers were aggregated and shuffled, and this combined dataset was partitioned into a training set (70%), a validation set (10%), and a test set (20%). The validation set was utilized for hyperparameter tuning. We employ Binary Cross Entropy with Logits Loss as the loss function during training. To address the data imbalance caused by an excessive number of non–change-point samples, we adopt the undersampling strategy, randomly removing samples without change points so that the ratio between change-point and non–change-point samples is approximately 4:6.

On the inbound traffic data, the trained model achieves 89.8% accuracy on the training set and 86.4% on the testing set for change-point identification. On the outbound traffic data, the model achieves 89.9% training accuracy and 88.6% testing accuracy. The corresponding ROC curves and AUC values for the testing sets are presented in Figure 3. With AUC values around 0.9, our model demonstrates a strong ability to accurately identify change points.

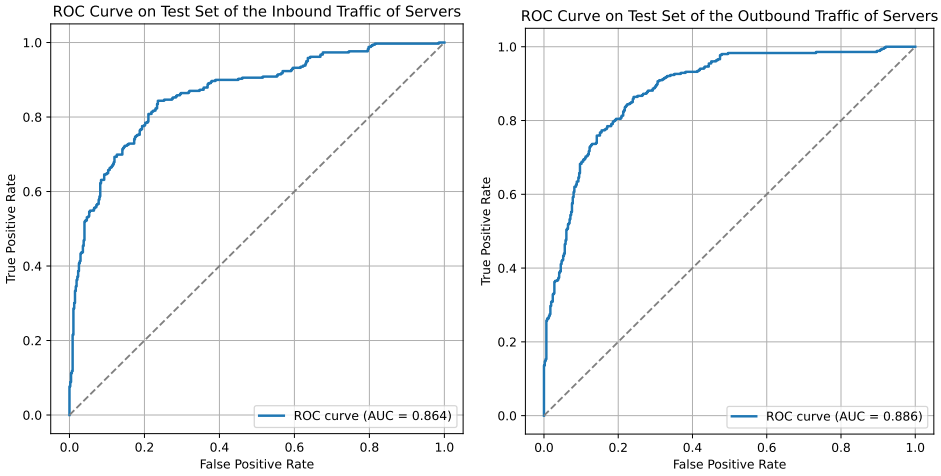


Figure 3: ROC curves for change point detection accuracy on the test sets of server inbound and outbound traffic data.

For comparison, we apply the fully functional method (FFM), the FPCA-based method (TEV), and kernel-based method (KLCPD) to detect change points on the real-world server traffic data. In addition, we include a simple neural baseline, denoted as FENet (CNN), which combines the functional embedding induced by the v-matrix with a convolutional neural network to capture temporal patterns. Since the theoretical development of FENet in the previous sections is based on a MLP architecture, the FENet (MLP) reported in Table 1 serves as the basic version of our method. In contrast, FENet (CNN) serves as an extension that replaces the MLP with a CNN while preserving the same functional embedding structure. The quantitative results are summarized in Table 1.

We report multiple evaluation metrics to provide a comprehensive assessment of detection performance. Specifically, Accuracy measures the overall classification correctness, TPR (True Positive Rate) reflects the ability to correctly identify true change points, and FAR (False Alarm Rate) evaluates the frequency of false detections. Precision is defined as $\mathrm{Precision} = \mathrm{TP}/(\mathrm{TP}+\mathrm{FP})$, which measures the reliability of detected change points. The F1 score is defined as $\mathrm{F1} = 2 \times \mathrm{Precision} \times \mathrm{TPR}/(\mathrm{Precision}+\mathrm{TPR})$, which balances Precision and TPR. Finally, AUC summarizes the overall ranking performance across different thresholds.

From Table 1, FENet consistently achieves the best overall performance across multiple evaluation metrics on both inbound and outbound datasets, indicating strong classification and ranking performance. While the TPR of FENet is slightly lower than that of FFM and TEV-90, this is accompanied by a substantially lower FAR. This reflects a more conservative and reliable detection behavior, avoiding excessive false alarms that are evident in the competing methods. Compared with the neural baseline KLCPD, FENet exhibits more balanced performance, achieving higher accuracy while maintaining a desirable level of false alarm control. The improved performance of FENet (CNN) demonstrates that the proposed functional embedding provides a strong and effective representation for change-point detection, which can be combined with different downstream architectures.

We further evaluate FENet (MLP) on the inbound traffic data under increased class imbalance and corrupted training labels. We retain the original 70/10/20 split and keep the validation and test sets, network architecture, training hyperparameters, and threshold-selection procedure fixed.

We consider positive-to-negative training ratios of 2:8 and 1:9. All positive training samples are retained in the 2:8 setting. For the 1:9 setting, the limited number of negative samples requires a deterministic subset of positives together with nearly all available negatives. Separately, we randomly flip 5% and 10% of the training labels, flipping equal numbers from each class to preserve the class counts. The labels flipped at 5% form a subset of those flipped at 10%. Each condition is evaluated with the same three model-initialization seeds (101, 202, and 303).

Table 1: Comparison of different change-point detection methods on inbound and outbound server traffic datasets across multiple evaluation metrics.

Type	Method	Accuracy	TPR	FAR	Precision	F1	AUC
Inbound	FENet (MLP)	0.790	0.785	0.206	0.731	0.757	0.864
	FENet (CNN)	0.840	0.791	0.124	0.820	0.805	0.899
	FFM	0.565	0.976	0.719	0.484	0.647	0.705
	TEV-85	0.580	0.985	0.701	0.493	0.657	0.893
	TEV-90	0.576	0.985	0.707	0.491	0.655	0.888
	TEV-95	0.566	0.985	0.723	0.485	0.650	0.889
	KLCPD	0.785	0.699	0.154	0.765	0.730	0.864
Outbound	FENet (MLP)	0.814	0.768	0.151	0.795	0.781	0.886
	FENet (CNN)	0.843	0.867	0.175	0.791	0.827	0.914
	FFM	0.614	0.972	0.657	0.529	0.685	0.753
	TEV-85	0.607	0.974	0.672	0.532	0.681	0.874
	TEV-90	0.602	0.974	0.681	0.520	0.679	0.871
	TEV-95	0.600	0.974	0.683	0.520	0.679	0.869
	KLCPD	0.808	0.768	0.162	0.783	0.775	0.865

Table 2: Sensitivity of FENet (MLP) to training class imbalance and label noise on inbound traffic data. Entries report means with standard deviations in parentheses over three initialization seeds.

Training condition	Accuracy	TPR	FAR	Precision	F1	AUC
Original undersampling	0.7903 (0.0217)	0.7591 (0.0586)	0.1874 (0.0711)	0.7489 (0.0587)	0.7510 (0.0153)	0.8662 (0.0070)
Training ratio 2:8	0.7711 (0.0332)	0.7099 (0.0325)	0.1853 (0.0794)	0.7394 (0.0725)	0.7219 (0.0198)	0.8458 (0.0107)
Training ratio 1:9	0.7662 (0.0114)	0.6667 (0.1161)	0.1628 (0.1004)	0.7608 (0.0824)	0.7013 (0.0281)	0.8156 (0.0543)
5% label flips	0.7805 (0.0099)	0.6922 (0.0567)	0.1565 (0.0237)	0.7603 (0.0133)	0.7235 (0.0256)	0.7965 (0.0546)
10% label flips	0.7735 (0.0102)	0.6716 (0.0264)	0.1537 (0.0036)	0.7571 (0.0065)	0.7117 (0.0171)	0.7977 (0.0147)

Table 2 shows modest reductions in accuracy and F1 under both perturbations, accompanied by lower TPR and AUC. These results indicate that FENet remains effective beyond the original undersampling ratio and clean training labels, although stronger imbalance and label corruption reduce its sensitivity.

Furthermore, we employ the trained single change point classifier to identify multiple change points in a long-sequence functional time series. Specifically, we consider the 30-day outbound traffic data of each server and examine whether one or more change points occur, as well as the corresponding times of occurrence. Four representative servers are presented in Figure 4, where the red solid lines and their associated coordinate values indicate the detected change-point positions.

For Server 66, a consistent outbound traffic pattern is observed during the first 18 days, characterized by periodic data transmissions within the first five minutes of each hour. After day 18, this pattern persists but with reduced transmission intensity, suggesting a decline in the volume of regular data collection by the external database. In the case of Server 71, the outbound traffic remains largely stable, except for days 3 to 6, where the traffic volume exceeds the baseline level, suggesting continuous data extraction from an external source during this period. Server 82 exemplifies a situation in which external access occurs exclusively on weekdays, with no activity observed during weekends. Given that the detection window of our classifier is set to seven days, the proposed multi-change-point detection algorithm successfully identifies each weekend transition as a distinct change point.

These empirical findings demonstrate that the proposed method effectively captures temporal shifts in server traffic patterns, thereby assisting system administrators in promptly detecting operational anomalies and determining whether manual intervention is warranted. Similar detection results were observed for the majority of servers, indicating the robustness and generalizability of the proposed approach across diverse operational conditions. By modeling each day's traffic as a

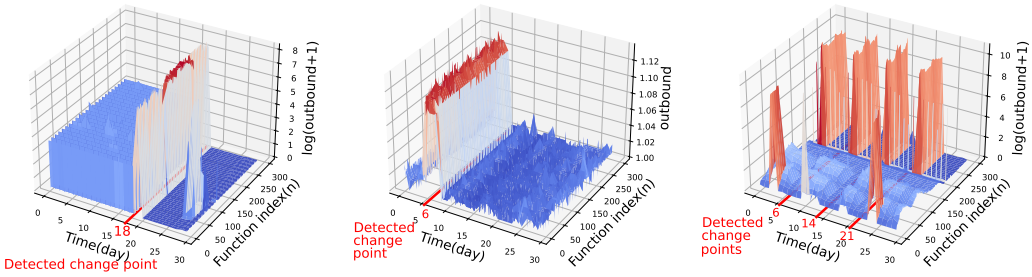


Figure 4: Detection of change points in the outbound traffic of different servers. The red lines indicate the detected change points. From left to right, the panels correspond to Servers 66, 71 and 82.

functional observation, the method mitigates the influence of transient fluctuations and noise, enabling stable and interpretable identification of meaningful structural changes. In practice, this framework can be seamlessly integrated into real-time monitoring systems, providing an automated early-warning mechanism for large-scale computing infrastructures. Beyond network traffic data, the same methodology can be extended to other types of functional signals, such as CPU utilization, memory usage, or temperature readings, offering a unified and scalable solution for dynamic system diagnostics and anomaly management.

5 Conclusion and Discussion

In this paper, we propose a deep learning framework for change-point detection in functional time series. We theoretically show that, under suitable choices of network architecture and activation functions, the proposed FENet can exactly represent the FCUSUM statistic, enabling the detection threshold λ to be learned within a supervised learning framework. Theoretical results, simulation studies, and empirical analyses collectively demonstrate the effectiveness of the proposed method in both detection accuracy and change-point localization. We also evaluate FENet on an additional publicly available real-world dataset; the corresponding code and experimental details are available in our public code repository.

Several directions remain for future work. In practice, functional time series are often multivariate, reflecting simultaneous changes across multiple operational indicators. The current study treats each indicator independently during labeling and training, which may increase the burden on system operators. Extending the proposed framework to multivariate functional data therefore constitutes an important direction for future research. Moreover, functional observations may be collected sparsely or at irregular time points. Adapting FENet to analyze sparse and irregularly sampled functional time series is another promising direction for future work.

Acknowledgments and Disclosure of Funding

The authors thank Chenming Qiu, Gang Xu, Qihan Tang, Yunkai Wang, and Huiying Zhang for their efforts in collecting and organizing the raw data sets used in this study. The authors also thank Yixin Che, Yanxin Dong, Zerui Zhao, and Gaofeng Zheng for their assistance in data labeling and preprocessing. The authors are grateful to Piotr Fryzlewicz for his helpful comments and suggestions on an early draft of this paper.

Dr. You's research was supported by the National Natural Science Foundation of China (NSFC, No. 11971291) and the Innovative Research Team of Shanghai University of Finance and Economics. Dr. Tang's work is supported in part by the National Key Research and Development Program of China (No. 2024YFB2705400). Dr. Cao's research is supported by a Discovery Grant (RGPIN-2023-04057) from the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Canada Research Chair program.

References

John AD Aston and Claudia Kirch. Detecting and estimating changes in dependent functional data. *Journal of Multivariate Analysis*, 109:204–220, 2012.

Alexander Aue and Anne Van Delft. Testing for stationarity of functional time series in the frequency domain. *The Annals of Statistics*, 48(5):2505–2547, 2020.

Alexander Aue, Robertas Gabrys, Lajos Horváth, and Piotr Kokoszka. Estimation of a change-point in the mean function of functional data. *Journal of Multivariate Analysis*, 100(10):2254–2269, 2009.

Alexander Aue, Siegfried Hörmann, Lajos Horváth, and Marie Hušková. Dependent functional linear models with applications to monitoring structural change. *Statistica Sinica*, pages 1043–1073, 2014.

Alexander Aue, Lajos Horváth, and Daniel F. Pellatt. Functional generalized autoregressive conditional heteroskedasticity. *Journal of Time Series Analysis*, 38(1):3–21, 2017.

Alexander Aue, Gregory Rice, and Ozan Sönmez. Detecting and dating structural breaks in functional data without dimension reduction. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 80(3):509–529, 2018.

Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 20(63):1–17, 2019.

István Berkes, Robertas Gabrys, Lajos Horváth, and Piotr Kokoszka. Detecting changes in the mean of functional observations. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 71(5):927–946, 2009.

Philippe C Besse, Hervé Cardot, and David B Stephenson. Autoregressive forecasting of some functional climatic variations. *Scandinavian Journal of Statistics*, 27(4):673–687, 2000.

Wei-Cheng Chang, Chun-Liang Li, Yiming Yang, and Barnabás Póczos. Kernel change-point detection with auxiliary deep generative models. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=r1GbfhRqF7`.

Rick Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

Birte Eichinger and Claudia Kirch. A mosum procedure for the estimation of multiple random change points. *Bernoulli*, pages 526–564, 2018.

Frédéric Ferraty and Philippe Vieu. *Nonparametric functional data analysis: theory and practice*. Springer, 2006.

Oleksandr Gromenko, Piotr Kokoszka, and Matthew Reimherr. Detection of change in the spatiotemporal mean function. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 79 (1):29–50, 2017.

Muktesh Gupta, Rajesh Wadhvani, and Akhtar Rasool. Real-time change-point detection: A deep neural network-based adaptive approach for detecting changes in multivariate time series data. *Expert Systems with Applications*, 209:118260, 2022.

Siegfried Hörmann and P Kokoszka. Weakly dependent functional data. *The Annals of Statistics*, 38 (3):1845–1884, 2010.

Lajos Horváth and Piotr Kokoszka. *Inference for functional data with applications*, volume 200. Springer Science & Business Media, 2012.

Lajos Horváth, Piotr Kokoszka, and Ron Reeder. Estimation of the mean of functional time series and a two-sample problem. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 75(1):103–122, 2013.

Lajos Horváth, Piotr Kokoszka, and Gregory Rice. Testing stationarity of functional time series. *Journal of Econometrics*, 179(1):66–82, 2014.

Moritz Jirak. On weak invariance principles for sums of dependent random functionals. *Statistics & Probability Letters*, 83(10):2291–2296, 2013.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL `https://arxiv.org/abs/1412.6980`.

Francesco Lattari, Alessio Rucci, and Matteo Matteucci. A deep learning approach for change points detection in insar time series. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–16, 2022.

Junghwan Lee, Yao Xie, and Xiuyuan Cheng. Training neural networks for sequential change-point detection. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

Degui Li, Peter M Robinson, and Han Lin Shang. Nonstationary fractionally integrated functional time series. *Bernoulli*, 29(2):1505–1526, 2023.

Degui Li, Runze Li, and Han Lin Shang. Detection and estimation of structural breaks in high-dimensional functional time series. *The Annals of Statistics*, 52(4):1716–1740, 2024a.

Jie Li, Paul Fearnhead, Piotr Fryzlewicz, and Tengyao Wang. Automatic change-point detection in time series via deep learning. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 86(2):273–285, 2024b.

Arnaud Liehrmann, Guillem Rigaill, and Toby Dylan Hocking. Increased peak detection accuracy in over-dispersed chip-seq data with supervised segmentation models. *BMC bioinformatics*, 22(1): 323, 2021.

Mehryar Mohri, A Rostamizadeh, and A Talwalkar. Foundations of machine learning, ser. adaptive computation and machine learning, 2012.

Kenta Oono and Taiji Suzuki. Approximation and non-parametric estimation of ResNet-type convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4922–4931. PMLR, 2019. URL `https://proceedings.mlr.press/v97/oono19a.html`.

Victor M Panaretos and Shahin Tavakoli. Fourier analysis of stationary time series in function space. *The Annals of Statistics*, 41(2):568–603, 2013.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

James O Ramsay and Bernard W Silverman. *Functional data analysis*. Springer, 2005.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Olimjon Sharipov, Johannes Tewes, and Martin Wendler. Sequential block bootstrap in a hilbert space with application to change point analysis. *Canadian Journal of Statistics*, 44(3):300–322, 2016.

Leonid Torgovitski. Detecting changes in hilbert space data based on “repeated" and change-aligned principal components. *arXiv preprint arXiv:1509.07409*, 2015.

Xianyang Zhang, Xiaofeng Shao, Katharine Hayhoe, and Donald J Wuebbles. Testing the structural stability of temporally dependent functional observations and application to climate projections. *Electronic Journal of Statistics*, 5:1765–1796, 2011.

A Additional Theoretical Results and Proofs

A.1 Additional Theoretical Results

The following Theorem 4 and Corollary 1 provide theoretical insights into controlling the mis-classification error of the FCUSUM classifier. Before proceeding with the theory, we first define a a positive definite and symmetric Hilbert–Schmidt integral operator c_ϵ on $L^2[0,1]$, given by

$$c_\epsilon(f)(t) = \int C_\epsilon(t,s)f(s)ds, \tag{8}$$

where $C_\epsilon(s,s') = \sum_{l=-\infty}^{\infty} \text{cov}\{\epsilon_0(s), \epsilon_l(s')\}$ is the long-run covariance kernel of the error sequences $(\epsilon_t, t \in \mathbb{Z})$. We further define a non-increasing sequence of non-negative eigenvalues $(\lambda_l : l \in \mathbb{N})$ and a corresponding orthonormal basis of eigenfunctions $(\phi_l(s) : l \in \mathbb{N})$ satisfying $c_\epsilon(\phi_l)(s) = \lambda_l\phi_l(s)$.

Theorem 4 *Under Assumption 1, [i] if $m = 0$, for sufficiently large K, we have*

$$P(||U(\tilde{\mathbf{x}})||_\infty > \lambda) \leq 2e^{-2\lambda/\lambda_1 K} \tag{9}$$

as $T \to \infty$, where λ_1 is the largest eigenvalues of c_ϵ defined on Equation (8)*.*
[ii] if $m \geq 1$, then $||U(\tilde{\mathbf{x}})||_\infty \xrightarrow{P} \infty$, as $T \to \infty$.

Remark 2 *In the absence of a change point, the asymptotic behavior of $||U(\tilde{\mathbf{x}})||_\infty$ is notably contingent on that of the partial sum process of the error terms $\epsilon_t(s)$, which involves the long-run covariance kernel C_ϵ. C_ϵ is a well-defined component of $L^2[0,1]^2$, serving as the converging covariance kernel of $\sqrt{T}$ times the centered sample mean. This kernel was initially explored in Hörmann and Kokoszka [2010], and subsequently investigated in Panaretos and Tavakoli [2013] within the domain of spectral analysis of functional time series, and in Horváth et al. [2013] in the context of the functional two-sample problem. c_ϵ is defined to be a positive definite and symmetric Hilbert–Schmidt integral operator on $L^2[0,1]$, with its eigenvalues determining the asymptotic distribution of $||U(\tilde{\mathbf{x}})||_\infty$.*

The following corollary controls the mis-classification risk of a functional FCUSUM classifier:

Corollary 1 *Define $Y = \mathbb{I}\{m \neq 0\}$, under Assumption 1, for sufficiently large K and $\lambda = K^2$, the classifier h_λ satisfies*

$$\mathbb{P}(h_\lambda(\tilde{\mathbf{x}}) \neq Y) \leq 2e^{-2K/\lambda_1},$$

when $T \to \infty$, and λ_1 is the largest eigenvalues of c_ϵ defined on Equation (8)*.*

We next extend the FCUSUM representation to convolutional networks. The following theorem shows that a depthwise CNN built on the functional embedding can reproduce the FCUSUM decision rule exactly.

Theorem 5 (Exact CNN realization) *Let $\mathcal{H}^{\text{CNN}}_{1,T-1,T}$ be the class of binary classifiers consisting of one depthwise Conv1D layer with $T-1$ input and output channels, $T-1$ groups, kernel size T, stride 1, no padding, and trainable kernels and biases. The layer is followed by ReLU, an affine readout, and the decision rule $\mathbb{I}\{g > 0\}$ for readout score g. For every $\lambda > 0$,*

$$h^{\text{FCUSUM}}_\lambda \in \mathcal{H}^{\text{CNN}}_{1,T-1,T}.$$

By Theorem 5, the convolutional class contains the FCUSUM classifier as a special case. This allows us to use FCUSUM as a comparator in the following generalization bound.

Theorem 6 (Generalization bound for convolutional FENet) *Let $\widehat{h}_{\text{CNN}}$ be an empirical risk minimizer over $\mathcal{H}^{\text{CNN}}_{1,T-1,T}$ under the 0–1 loss, based on N independent copies of $(\widetilde{\mathbf{X}}, Y)$. Define the population risk $R(h) = \mathbb{P}\{h(\widetilde{\mathbf{X}}) \neq Y\}$ and $d_T = \text{VCdim}(\mathcal{H}^{\text{CNN}}_{1,T-1,T})$. If $N \geq \max\{2, d_T\}$, there exists a universal constant $C_0 > 0$ such that, for any fixed $\lambda > 0$ and every $\delta \in (0,1)$, with probability at least $1-\delta$,*

$$R(\widehat{h}_{\text{CNN}}) \leq R(h^{\text{FCUSUM}}_\lambda) + C_0\sqrt{\frac{T^2 \log T \log N + \log(1/\delta)}{N}}.$$

Theorem 6 bounds the population risk of the empirical risk minimizer by the FCUSUM risk plus an estimation term that vanishes as N increases for fixed T. The proofs of both theorems are given in Appendix A.2.

A.2 Proofs of Theoretical Results

In this section, we provide the proofs of the necessary lemmas and asymptotic theorems presented in the main paper.

Proof of Lemma 1: In this proof, our primary goal is to clarify the definition of the FCUSUM statistic and the mechanism of embedding layer in FENet.

On the one hand, for

$$U_t(s) = \frac{1}{\sqrt{T}}(\sum_{j=1}^{t} X_j(s) - \frac{t}{T}\sum_{j=1}^{T} X_j(s)),$$

We have

$$\begin{aligned}
M_t =& ||U_t(s)||^2 = \int_0^1 |U_t(s)|^2 ds \\
=& \int_0^1 \frac{1}{T}[\sum_{j=1}^{t} X_j(s) - \frac{t}{T}\sum_{j=1}^{T} X_j(s)]^2 ds \\
=& \int_0^1 \frac{1}{T}[\frac{T-t}{T}\sum_{j=1}^{t} X_j(s) - \frac{t}{T}\sum_{j=t+1}^{T} X_j(s)]^2 ds \\
=& \frac{1}{T}\int_0^1 (\frac{T-t}{T})^2(\sum_{j=1}^{t} X_j(s))^2 - 2\frac{t}{T}(\frac{T-t}{T})(\sum_{j=1}^{t} X_j(s))(\sum_{j=t+1}^{T} X_j(s)) \\
&+ \frac{t^2}{T^2}(\sum_{j=t+1}^{T} X_j(s))^2 ds \\
=& \frac{1}{T}[\int_0^1 (\frac{T-t}{T})^2(\sum_{j=1}^{t} X_j(s))^2 ds - 2\frac{t}{T}(\frac{T-t}{T})\int_0^1 (\sum_{j=1}^{t} X_j(s))(\sum_{j=t+1}^{T} X_j(s)) ds \\
&+ \frac{t^2}{T^2}\int_0^1 (\sum_{j=t+1}^{T} X_j(s))^2 ds].
\end{aligned}$$

On the other hand, define $\mathbf{v}_t = ((\frac{1}{T} - \frac{t}{T^2}) \cdot \mathbf{1}_t, -\frac{t}{T^2} \cdot \mathbf{1}_{T-t})^\top$, $\mathbf{v}'_t = ((1 - \frac{t}{T}) \cdot \mathbf{1}_t, -\frac{t}{T} \cdot \mathbf{1}_{T-t})^\top$ for $t = 1, \cdots, T-1$ and

$$\langle \mathbf{X}(s), \mathbf{X}(s) \rangle = \begin{pmatrix} \langle X_1(s), X_1(s) \rangle & \langle X_1(s), X_2(s) \rangle & \cdots & \langle X_1(s), X_T(s) \rangle \\ \langle X_2(s), X_1(s) \rangle & \langle X_2(s), X_2(s) \rangle & \cdots & \langle X_2(s), X_T(s) \rangle \\ \cdots & \cdots & \cdots & \cdots \\ \langle X_T(s), X_1(s) \rangle & \langle X_T(s), X_2(s) \rangle & \cdots & \langle X_T(s), X_T(s) \rangle \end{pmatrix},$$

where

$$\langle X_i(s), X_j(s) \rangle = \int_0^1 X_i(s) X_j(s) ds,$$

then

$$
\begin{aligned}
&\mathbf{v}_t^{'\top}\langle \mathbf{X}(s), \mathbf{X}(s)\rangle \mathbf{v}_t \\
=&(1-\tfrac{t}{T}, 1-\tfrac{t}{T}, \cdots, -\tfrac{t}{T}, -\tfrac{t}{T}) \begin{pmatrix} \langle X_1(s), X_1(s)\rangle & \cdots & \langle X_1(s), X_T(s)\rangle \\ \vdots & \ddots & \vdots \\ \langle X_T(s), X_1(s)\rangle & \cdots & \langle X_T(s), X_T(s)\rangle \end{pmatrix} \begin{pmatrix} \frac{1}{T} - \frac{t}{T^2} \\ \vdots \\ -\frac{t}{T^2} \end{pmatrix} \\
=&(1-\tfrac{t}{T}, \cdots, -\tfrac{t}{T}) \begin{pmatrix} \sum_{j=1}^{t}(\frac{1}{T} - \frac{t}{T^2})\langle X_1(s), X_j(s)\rangle + \sum_{j=t+1}^{T}(-\frac{t}{T^2})\langle X_1(s), X_j(s)\rangle \\ \vdots \\ \sum_{j=1}^{t}(\frac{1}{T} - \frac{t}{T^2})\langle X_T(s), X_j(s)\rangle + \sum_{j=t+1}^{T}(-\frac{t}{T^2})\langle X_T(s), X_j(s)\rangle \end{pmatrix} \\
=&\frac{1}{T}\Bigg\{(1-\tfrac{t}{T})^2 \sum_{i=1}^{t}\sum_{j=1}^{t}\langle X_i(s), X_j(s)\rangle + 2(-\tfrac{t}{T})(1-\tfrac{t}{T}) \sum_{i=1}^{t}\sum_{j=t+1}^{T}\langle X_i(s), X_j(s)\rangle \\
&\quad + (\tfrac{t}{T})^2 \sum_{i=t+1}^{T}\sum_{j=t+1}^{T}\langle X_i(s), X_j(s)\rangle\Bigg\} \\
=&M_t
\end{aligned}
$$

Define $\tilde{\mathbf{x}}_t = \langle \mathbf{X}(s), \mathbf{X}(s)\rangle \mathbf{v}_t$, then we have

$$
\mathbf{M} = \begin{pmatrix} M_1 \\ M_2 \\ \cdots \\ M_{T-1} \end{pmatrix} = \begin{pmatrix} \mathbf{v}_1'\langle \mathbf{X}(s), \mathbf{X}(s)\rangle \mathbf{v}_1 \\ \mathbf{v}_2'\langle \mathbf{X}(s), \mathbf{X}(s)\rangle \mathbf{v}_2 \\ \cdots \\ \mathbf{v}_{T-1}'\langle \mathbf{X}(s), \mathbf{X}(s)\rangle \mathbf{v}_{T-1} \end{pmatrix} = \begin{pmatrix} \mathbf{v}_1'\tilde{\mathbf{x}}_1 \\ \mathbf{v}_2'\tilde{\mathbf{x}}_2 \\ \cdots \\ \mathbf{v}_{T-1}'\tilde{\mathbf{x}}_{T-1} \end{pmatrix} := U(\tilde{\mathbf{x}})
$$

and

$$
\tilde{\mathbf{x}} = (\tilde{\mathbf{x}}_1^\top, \cdots, \tilde{\mathbf{x}}_{T-1}^\top)^\top.
$$

Then

$$
\begin{aligned}
h_\lambda(\mathbf{X}(s)) =& \mathbb{I}\{||\mathbf{M}||_\infty > \lambda\} \\
=& \mathbb{I}\{\max_{t\in[T-1]} M_t > \lambda\} \\
=& \mathbb{I}\{\max_{t\in[T-1]} \mathbf{v}_t'\langle \mathbf{X}(s), \mathbf{X}(s)\rangle \mathbf{v}_t > \lambda\} \\
=& \mathbb{I}\{\max_{t\in[T-1]} \mathbf{v}_t'\tilde{\mathbf{x}}_t > \lambda\}
\end{aligned}
$$

Proof of Theorem 1: Define matrix

$$
(\mathbf{W}_0)_{(T-1)\times T(T-1)} = \begin{pmatrix} \mathbf{v}_1^{'\top} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{v}_2^{'\top} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{v}_3^{'\top} & \cdots & \mathbf{0} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{v}_{T-1}^{'\top} \end{pmatrix}
$$

and $\mathbf{W}_1 := \mathbf{1}_{T-1}$, $\mathbf{a}_1 := -\lambda \mathbf{1}_{T-1}$ and $a_2 = 0$. Then $h(\tilde{\mathbf{x}}) := \sigma_{a_2}^* W_1 \sigma_{\mathbf{a}_1} W_0 \tilde{\mathbf{x}}$ can be rewritten as

$$
h(\tilde{\mathbf{x}}) = \mathbb{I}\{\sum_{t=1}^{T-1}(\mathbf{v}_t^{'\top}\tilde{\mathbf{x}}_t - \lambda)_+ > a_2\} = \mathbb{I}\{\max_{t\in[T-1]} \mathbf{v}_t^{'\top}\tilde{\mathbf{x}}_t > \lambda\} = h_\lambda(\tilde{\mathbf{x}})
$$

as desired.

Lemma 2 *Under model 1, Assumption 1 and* $m = 0$*,*

$$
||U(\tilde{\mathbf{x}})||_\infty \xrightarrow{d} \sup_{0\le x\le 1} \sum_{l=1}^{\infty} \lambda_l B_l^2(x) \quad (T \to \infty),
$$

where $(B_l : l \in \mathbb{N})$ *are IID standard Brownian bridges defined on* $[0, 1]$.

Proof of Lemma 2: Let

$$C'_T(t,s) = \frac{1}{\sqrt{T}} \sum_{i=1}^{t} \epsilon_i(s) = \frac{1}{\sqrt{T}} \sum_{i=1}^{\lfloor Tx \rfloor} \epsilon_i(s) = C_T(x,s)$$

where $\lfloor Tx \rfloor = t$. Then, by Theorem 1.2 of Jirak [2013], under Assumption 1, there exists a sequence of Gaussian processes denoted as $\Gamma_T(x,s) : T \in \mathbb{N}, x, s \in [0,1]$ such that $\mathbb{E}[\Gamma_T(x,s)] = 0$, $\mathbb{E}[\Gamma_T(x,s)\Gamma_T(x',s')] = \min\{x,x'\}C_\epsilon(t,t')$ and

$$\sup_{0\le x\le 1} \int (C_T(x,s) - \Gamma_T(x,s))^2 ds = o_p(1). \tag{10}$$

When $m = 0$, we can assume without loss of generality (WLOG) that $\mu_0 = 0$ in model Equation (1). In this scenario,

$$\begin{aligned} U_t(s) &= \frac{1}{\sqrt{T}} \Big\{ \sum_{j=1}^{t} X_j(s) - \frac{t}{T} \sum_{j=1}^{T} X_j(s) \Big\} \\ &= C'_T(t,s) - \frac{t}{T} C'_T(T,s) \\ &= C_T(\lfloor t/T \rfloor, s) - \frac{t}{T} C_T(1,s). \end{aligned}$$

Furthermore, it can be demonstrated through moment arguments that

$$||U(\tilde{\mathbf{x}})||_\infty = \sup_{0\le x\le 1} ||C_T(x,s) - xC_T(1,s)||^2 + o_p(1). \tag{11}$$

Letting $\Gamma^0_T(x,s) = \Gamma_T(x,s) - x\Gamma_T(1,s)$, it follows that

$$\begin{aligned} &\sup_{0\le x\le 1} ||C_T(x,s) - xC_T(1,s)||^2 \\ &= \sup_{0\le x\le 1} ||\Gamma^0_T(x,s)||^2 + \sup_{0\le x\le 1} ||C_T(x,s) - xC_T(1,s)||^2 - \sup_{0\le x\le 1} ||\Gamma^0_T(x,s)||^2. \end{aligned} \tag{12}$$

The conjunction of the triangle inequality with Equation (10) leads to the inference that

$$\sup_{0\le x\le 1} ||C_T(x,s) - \frac{x}{T} C_T(1,s)||^2 - \sup_{0\le x\le 1} ||\Gamma^0_T(x,s)||^2 = o_p(1),$$

and this along with Equation (11) and Equation (12) yield

$$||U(\tilde{\mathbf{x}})||_\infty = \sup_{0\le x\le 1} ||\Gamma^0_T(x,s)||^2 + o_p(1). \tag{13}$$

By performing direct calculations based on the definition of $\Gamma_T(x,s)$, it can be illustrated that

$$\mathbb{E}[\Gamma^0_T(x,s)\Gamma^0_T(x',s')] = (\min\{x,x'\} - xx')C_\epsilon(t,t').$$

Therefore, for each T, the Gaussian process $\Gamma^0_T(x,s)$ is distributed identically to

$$\sum_{l=1}^{\infty} \sqrt{\lambda_l} B_l(x) \phi_l(s)$$

where $B_l(s) : l \in \mathbb{N}$ are IID standard Brownian bridges defined on $[0,1]$ and λ_l is the lth eigenvalues of c_ϵ with a corresponding orthonormal basis of eigenfunction $\phi_l(s)$ satisfying

$$c_\epsilon(\phi_l)(s) = \int C_\epsilon(s,s')\phi_l(s')ds' = \lambda_l \phi_l(s)$$

and

$$C_\epsilon(s,s') = \sum_{l=-\infty}^{\infty} \text{cov}\{\epsilon_0(s), \epsilon_l(s')\}$$

is the long-run covariance kernel of the error sequences $(\epsilon_t, t \in \mathbb{Z})$. c_1, c_2 are constants that depend on the smoothness of the process covariance and the rate of decay of eigenvalues.

It follows that, for all T,

$$\sup_{0\le x\le 1}||\Gamma_T^0(x,s)||^2 \overset{d}{=} \sup_{0\le x\le 1}\sum_{l=1}^{\infty}\lambda_l B_l^2(x),$$

which, in accordance with Equation (13), leads to the theorem.

Proof of Theorem 4: [i] From Lemma 2, we have

$$||U(\tilde{\mathbf{x}})||_\infty \xrightarrow{d} \sup_{0\le x\le 1}\sum_{l=1}^{\infty}\lambda_l B_l^2(x)$$

as $T\to\infty$. By Assumption 1, we have $\sum_{l=1}^{\infty}\lambda_l<\infty$, then we have

$$\mathbb{E}(\sum_{l=1}^{\infty}\lambda_l B_l^2(x)-\sum_{l=1}^{K}\lambda_l B_l^2(x))^2\to 0$$

as $K\to\infty$. We hence suppose that $||U(\tilde{\mathbf{x}})||_\infty\sim\sum_{l=1}^{K}\lambda_l B_l^2(x)$ for sufficiently large K.

As $\lambda_1>\lambda_2>\cdots>\lambda_K$ and $\mathbb{P}(\sup_{0\le x\le 1}|B_1(x)|<b)=\sum_{m=-\infty}^{\infty}(-1)^m e^{-2m^2\lambda b^2}$ from Durrett [2019], we have the following result:

$$\begin{aligned}
\mathbb{P}(||U(\tilde{\mathbf{x}})||_\infty\ge\lambda) &= \mathbb{P}(\sup_{0\le x\le 1}\sum_{l=1}^{K}\lambda_l B_l^2(x)\ge\lambda)\\
&\le \mathbb{P}(\sup_{0\le x\le 1}K\lambda_1 B_1^2(x)\ge\lambda)\\
&= 1-\mathbb{P}(\sup_{0\le x\le 1}|B_1(x)|\le\sqrt{\lambda/K\lambda_1})\\
&= 1-\sum_{m=-\infty}^{\infty}(-1)^m e^{-2m^2\lambda/K\lambda_1}\\
&\le 2e^{-2\lambda/\lambda_1 K}.
\end{aligned}$$

[ii] When $m=1$, letting

$$C_{T,t,\epsilon}^0(s)=\frac{1}{T}(\sum_{i=1}^{t}\epsilon_i(s)-\frac{t}{T}\sum_{t=1}^{T}\epsilon_i(s)),$$

one has that, under Equation (2),

$$\begin{aligned}
||U(\tilde{\mathbf{x}})||_\infty=\max_{1\le t<T}||U(t)||^2 \ge& ||U(t^*)||^2\\
=&||\frac{1}{T}(\sum_{i=1}^{t^*}X_i(s)-\frac{t^*}{T}\sum_{i=1}^{T}X_i(s))||^2\\
=&||\frac{1}{T}\sum_{i=1}^{t^*}(\mu(s)+\epsilon_i(s))-\frac{t^*}{T^2}\sum_{i=1}^{T}(\mu(s)+\delta(s)+\epsilon_i(s))||^2\\
=&||C_{T,t^*,\epsilon}^0(s)-\frac{(T-t^*)t^*}{T^2}\delta(s)||^2,
\end{aligned}$$

Moreover, by the triangle inequality,

$$||C_{T,t^*,\epsilon}^0(s)-\frac{(T-t^*)t^*}{T^2}\delta(s)||^2\ge||\frac{(T-t^*)t^*}{T^2}\delta(s)||^2-||C_{T,t^*,\epsilon}^0(s)||^2.$$

Equation (10) implies that $||C_{T,t^*,\epsilon}^0(s)||^2=O_p(1)$, and, since $t^*=\lfloor T\theta\rfloor$ for some $\theta\in(0,1)$,

$$||\frac{(T-t^*)t^*}{T^2}\delta(s)||^2\ge c_{1,T}\sqrt{T}\theta(1-\theta)||\delta(s)||^2\to\infty,\quad(T\to\infty),$$

as $c_{1,T}$ may be taken so that $c_{1,T} \to 1$, which implies the result.

Proof of Theorem 2: Given any $L \geq 1$ and $\mathbf{m} = (m_1, \cdots, m_L)^T$, let $m_0 := T$ and $m_{L+1} := 1$ and set $W^* = \sum_{r=1}^{L+1} m_{r-1} m_r$. Let $d := VC\dim(\mathcal{H}_{L,\mathbf{m}})$, then by Theorem 7 of Bartlett et al. [2019], we have $d = O(LW^* \log(W^*))$. Thus, by Corollary 3.4 of Mohri et al. [2012], for a constant $C > 0$, we have with probability at least $1 - \delta$ that

$$\mathbb{P}(h_{ERM}(\tilde{\mathbf{x}}) \neq Y|\mathcal{D}) \leq \min_{h \in \mathcal{H}_{L,m}} \mathbb{P}(h(\tilde{\mathbf{x}}) \neq Y) + \sqrt{\frac{8d\ln(2eN/d) + 8\ln(4/\delta)}{N}}. \tag{14}$$

Here, we have $L = 1$,$m = T - 1$,$W^* = T(T-1)^2 = O(T^3)$. In addition, since $h_\lambda^{CUSUM}(\tilde{\mathbf{x}}) \in \mathcal{H}_{1,T-1}$, we have $\min_{h \in \mathcal{H}_{1,T-1}} \leq \mathbb{P}(h_\lambda^{CUSUM}(\tilde{\mathbf{x}})) = \mathbb{P}(||U(\tilde{\mathbf{x}})||_\infty \leq \lambda) \leq 2e^{-2\lambda/\lambda_1 K}$. Substituting these bounds into Equation (14) we have

$$\mathbb{P}(h_{ERM}(\tilde{\mathbf{x}}) \neq Y|\mathcal{D}) \leq 2e^{-2\lambda/\lambda_1 K} + \sqrt{\frac{T^3\ln(T)\ln(N) + \ln(1/\delta)}{N}}. \tag{15}$$

Proof of Theorem 3: For simplicity of presentation, we focus on the case where n is a multiple of 4, so $\gamma = 1/2$. Define

$$I_0 := \{t : \mu_{t+T-1}(s) = \mu_t(s)\}, I_1 := \{t : \mu_{t+T-1}(s) \neq \mu_t(s)\}.$$

By Corollary 1 and a union bound $\lambda = K^2$, the event

$$\mathcal{I} = \{h_\lambda(\tilde{\mathbf{x}}) = k, \text{for all} \quad t \in I_i, i = 0, 1\}$$

has probability at least $1 - 2e^{-2K/\lambda_1}$ when $T^* \to \infty$. We work on the event $\mathcal{I}$ henceforth.

Denote $\Delta = 2C^2\rho^{-2}$. Since $\rho \geq 2\sqrt{2}C$, we have $\Delta < 1/4$. Note that for each $k \in [m]$, we have $\{t : t_{k-1} < t \leq t_k - T\} \cup \{t : t_k < t \leq t_{k+1} - T\} \subseteq I_0$ and $\{t : t_k - T + \Delta < t \leq t_k - \Delta\} \subseteq I_1$. Consequently, $\bar{L}_i$ defined in Algorithm 1 is below the threshold $\gamma = 1/2$ for all $t \in (t_{k-1} + T/2, t_k - T/2] \cup (t_k + T/2, t_{k+1} - T/2]$, monotonically increases for $t \in (t_k - T/2, t_k - \Delta]$ and monotonically decreases for $t \in (t_k + \Delta, t_k + T/2]$ and is above the threshold γ for $t \in (t_k - \Delta, t_k + \Delta]$. Thus, exactly one change-point, say $\hat{t}_k$, will be identified on $(t_{k-1} + T/2 < t, t_{k+1} - T/2]$ and $\hat{t}_k = \arg\max_{t \in (t_{k-1}+T/2 < t, t_{k+1}-T/2]} \bar{L}_t \in (t_k - \Delta, t_k + \Delta]$ as desired. Since the above holds for all $\gamma \in [m]$, the proof is complete.

Proof of Theorem 5. For channel t, choose the kernel $\mathbf{k}_t = \mathbf{v}_t'$ and bias $-\lambda$. Using the Conv1D cross-correlation convention, the full-length kernel produces the scalar

$$z_t = \mathbf{k}_t^\top \tilde{\mathbf{x}}_t - \lambda = M_t - \lambda.$$

After ReLU, choose all readout weights equal to one and the readout bias equal to zero. The resulting score is

$$g_\lambda(\widetilde{\mathbf{X}}) = \sum_{t=1}^{T-1} (M_t - \lambda)_+,$$

where $(a)_+ = \max\{a, 0\}$. This score is positive exactly when $\max_t M_t > \lambda$. Thresholding at zero therefore gives $h_\lambda^{\mathrm{FCUSUM}}$.

Proof of Theorem 6. The convolutional layer contains $T(T-1)$ kernel parameters and $T-1$ biases. The readout has $T-1$ weights and one bias, giving $W_T = T^2 + T - 1$ trainable parameters. Since the network has constant depth and piecewise-linear activations, Theorem 7 of Bartlett et al. [2019] yields

$$d_T \leq CW_T \log W_T = O(T^2 \log T).$$

The standard VC bound for empirical risk minimization, as in Corollary 3.4 of Mohri et al. [2012], implies that, with probability at least $1 - \delta$,

$$R(\widehat{h}_{\mathrm{CNN}}) \leq \inf_{h \in \mathcal{H}_{1,T-1,T}^{\mathrm{CNN}}} R(h) + C\sqrt{\frac{d_T \log(eN/d_T) + \log(1/\delta)}{N}}.$$

By Theorem 5, the infimum is at most $R(h_\lambda^{\mathrm{FCUSUM}})$. Substituting the bound on d_T and using $\log(eN/d_T) \leq \log(eN)$ proves the result, following the same argument as Theorem 2.

A complementary representation follows from Theorem 1 of Oono and Suzuki [2019]. Vectorize the embedding as $\tilde{\mathbf{x}} \in \mathbb{R}^{D_T}$, where $D_T = T(T-1)$, and let $\mathbf{a}_t$ equal $\mathbf{v}_t'$ in block t and zero elsewhere. Then

$$g_\lambda(\tilde{\mathbf{x}}) = \sum_{t=1}^{T-1} \sigma(\mathbf{a}_t^\top \tilde{\mathbf{x}} - \lambda)$$

is a block-sparse fully connected network with $T-1$ blocks, each of depth one and width one. Their theorem provides an exact ResNet-type CNN representation of this score, whose thresholded output also reproduces FCUSUM. This gives complementary support for convolutional representability, while the direct construction above yields the explicit parameter count used in Theorem 6.

B Implementation Details

All experiments are implemented in PyTorch [Paszke et al., 2019] and conducted on a single NVIDIA A100 GPU with 80GB memory, except for the computational benchmark in Experiment 5, which is conducted on an 8-core Apple M2 CPU with 8 GB RAM. The ADAM optimizer [Kingma and Ba, 2017] is used across all experiments. The longest running time among all experiments is less than 15 minutes. The specific training procedure can be found in Algorithm 1.

B.1 Algorithmic Framework

The general procedure for multiple change-point detection is summarized in Algorithm 1; see, for example, Eichinger and Kirch [2018] and Li et al. [2024b] for related formulations.

Algorithm 1 Algorithm for change-point localization

Input: Functional time series $\{X_t(s)\}_{t=1}^{T^*}$, size of sliding window T, a trained FENet classifier $\psi : \mathbb{R}^{T(T-1)\times 1}$, $\gamma > 0$.
Procedure
[1] Form $\mathbf{X}^*_{[i,T+i)}(s) := (X_i(s), \cdots, X_{T+i-1}(s))$ and compute $\tilde{\mathbf{x}}_i$, Let $L_i = \psi(\tilde{\mathbf{x}}_i)$ for all $i = 1, \cdots, T^* - T + 1$;
[2] Compute $\bar{L}_i = T^{-1} \sum_{j=i-T+1}^{i} L_j$ for $i = T, \cdots, T^* - T + 1$;
[3] Let $\{[a_1, b_1], \cdots, [a_{\hat{m}}, b_{\hat{m}}]\}$ be the set of all maximal segments such that $\bar{L}_i > \gamma$ for all $i \in [a_r, b_r]$, $r \in [\hat{m}]$;
[4] Compute $\hat{t}_r = \arg\max_{i \in [a_r, b_r]} \bar{L}_i$ for all $r \in [\hat{m}]$;
Output: Estimated change-points $\hat{t}_1, \cdots, \hat{t}_{\hat{m}}$.

B.2 Evaluation Metrics

This section provides formal definitions of the evaluation metrics used in our experiments to assess change-point detection and localization accuracy.

- P_m represents the estimation accuracy of the change-point number, defined by
$$P_m = \tilde{N}_{\text{test}} / N_{\text{test}},$$
where $\tilde{N}_{\text{test}}$ denote the number of test samples in which the correct number of change points is identified.
- NMI measures the amount of information shared between the predicted clusters and the true class labels, normalized to account for differing cluster sizes. It is defined as:
$$\text{NMI}(\hat{C}, C) = 2 \frac{I(\hat{C}, C)}{H(\hat{C}) + H(C)},$$
where $H(\hat{C})$ and $H(C)$ are the entropy of the predicted clusters and true labels and $I(\hat{C}, C)$ is the mutual information between the predicted cluster $\hat{C}$ and the true labels C defined by
$$I(\hat{C}, C) = \sum_{i=1}^{\hat{m}} \sum_{j=1}^{m} \left(\frac{|\hat{C}_i \bigcap C_j|}{T^*}\right) \log_2 \left(\frac{T^* |\hat{C}_i \bigcap C_j|}{|\hat{C}_i||C_j|}\right).$$

- Purity is a simple measure that evaluates the extent to which each cluster contains members from a single ground truth class. It is defined as the ratio of the number of correctly assigned instances to the total number of instances, and it is given by:

$$\text{Purity} = \frac{1}{T^*} \sum_{i=1}^{\hat{m}} \max_{1 \le j \le m} |\hat{C}_i \bigcap C_j|.$$

- The average Hausdorff distance(HD) is defined by

$$\text{HD} = \frac{1}{T^*} \max\{\mathcal{D}(\hat{C}, C), \mathcal{D}(C, \hat{C})\},$$

where $\mathcal{D}(A, B) = \sup_{a \in A} \inf_{b \in B} |a - b|$, represents the Hausdorff distance between sets A and B. In this case, $\hat{C}$ denotes the estimated change points, and C represents the actual change points.

- RMSE of change-point estimation is defined as

$$\text{RMSE} = \sqrt{1/\tilde{N}_{\text{test}} \sum_{i=1}^{\tilde{N}_{\text{test}}} \sum_{j=1}^{m} (\hat{t}_{ij} - t_{ij})^2}.$$

B.3 Labeling Procedure for Server Operational Data

For each server and each traffic direction, the five-minute measurements are organized into daily traffic profiles, with 288 observations in a complete day. Each daily profile is treated as one functional observation. We construct seven-day rolling windows with a one-day stride. Consequently, a complete 100-day sequence yields 94 windows per server and traffic direction.

A window is assigned a change-presence label if the operating profiles on at least two consecutive days within the window are clearly different from the profiles on the remaining days. The assessment concerns persistent differences in the daily operating pattern, including changes in level, shape, or variability. An anomaly confined to a single day is not treated as a change point, even when that day contains several intraday spikes. The rule specifies a visual assessment of daily profiles and does not impose an additional numerical threshold or an exact 48-hour duration requirement. The final change-point annotation results have been verified by specialized operations personnel.

C Extra Experiments

In the upcoming analysis, we aim to empirically explore the effectiveness of training a FENet for change-point detection. Building upon the insights gleaned from the preceding section, we will train the FENet and delve into the impact of different sample types and training data sizes on its performance. Comparisons will be drawn against the fully functional method (FFM) proposed by Aue et al. [2018] and the functional principle analysis (FPCA) introduced by Berkes et al. [2009].

We conduct Monte Carlo simulations to systematically assess the performance of the proposed method under a range of controlled functional change-point settings. The functional observations are generated from a piecewise-constant mean structure with additive stochastic innovations, allowing for a single change point in the mean function. To examine the impact of the structural complexity of the change function, we consider three distinct data-generating scenarios that differ in the choice and dimensionality of the underlying basis representations.

Specifically, Scenario 1 employs randomly generated basis functions to represent localized and irregular changes, serving as a benchmark for numerical perturbations. Scenarios 2 and 3 adopt Fourier basis expansions with low and moderate dimensions, respectively, thereby capturing increasingly smooth yet structurally richer changes in the mean function. This design enables us to evaluate the sensitivity of competing methods to both the smoothness and the effective dimensionality of the change structure. For each scenario, we further consider both independent and functional autoregressive innovations to assess robustness under different temporal dependence structures. Detailed specifications of the data-generating mechanisms, parameter configurations, and labeling procedures are provided in Appendix C.1.

C.1 Data Generating Process

We begin by describing the generation process of the functional curves defined by $X_t(s) = \mu_k(s)\mathbb{I}_{(t_k^*, t_{k+1}^*]}(t) + \epsilon_t(s)$ in Equation (1), for $k = 0, \cdots, m$. For simplicity, the initial mean curve $\mu_0(s)$ is assumed to be the zero function.

To investigate the influence of the break function's distribution and its magnitude relative to the innovations level, we define the following class of break functions $\mu_k(s) \in$

$$\Big\{c\mu^{(\xi)}(s) : c \in \mathbb{R}^+, \mu^{(\xi)}(s) = \sqrt{\frac{1}{\xi}} \sum_{l=1}^{\xi} v_l(s), \xi \in [1, D]\Big\}, \tag{16}$$

for $k = 1, \cdots, m$, where c serves as the signal-to-noise ratio (SNR) parameter, $\mu^{(\xi)}(s)$ denotes the average of one or more basis functions, and ξ is independent of the SNR. The basis functions $v_1(s), \cdots, v_D(s)$ are considered on the interval $[0, 1]$ under the following three scenarios.

Scenario 1: The basis functions are derived from a uniform distribution with $D = 21$.

Scenario 2: The basis functions are Fourier basis function with $D = 6$.

Scenario 3: The basis functions are Fourier basis function with $D = 21$.

To minimize the influence of the specific structure of the Fourier basis functions and the arrangement of $v_1(s), \cdots, v_D(s)$ on the effectiveness of the different methodologies, a random permutation denoted as π was implemented on $1, \cdots, D$ before each simulation iteration. Subsequently, the experiment was conducted as outlined earlier, employing the permuted sequence $v_{\pi(1)}, \cdots, v_{\pi(D)}$.

The three scenarios depict distinct changes in the mean function. Scenario 1 signifies numerical alterations, while Scenario 2 represents relatively smooth functional data with both changes in the range and shape of the mean function. On the other hand, Scenario 3 exemplifies more intricate functional data. The variances in data generation, as well as the varying number of change functions denoted by ξ, are depicted in Figure 5. The value of c is set to 1.

We consider two types of innovations. The first type consists of independent curves (IID), defined as $\epsilon_t(s) = \eta_t(s)$, for $t = 1, \cdots, T$, where $\eta_t(s) = \sum_{l=1}^{D} G_{t,l} v_l(s)$, and $G_{t,l}$ represent independent zero-mean Gaussian random variables with standard deviations $\sigma = (3^{-l} : l = 1, \cdots, D)$. This specification is designed to emulate the rapid decay pattern typically observed in the eigenvalues of both the covariance and long-run covariance operators. The second type involves functional auto-regressive (FAR) curves, which are commonly utilized to capture the serial correlation of functional data, as discussed in Besse et al. [2000]. First-order FAR processes were generated as $\epsilon_t(s) = \Psi\epsilon_{t-1}(s) + \eta_t(s)$ for $t = 1, \cdots, T$, with a burn-in period of 100 initial curves that were discarded. The operator was constructed as $\Psi = 0.5\Psi_0$, where the random operator Ψ_0 is a $D \times D$ matrix whose entries are independent, mean-zero normal random variables with standard deviations given by $\sigma\sigma^\top$, as in Aue et al. [2018].

We next describe the labeling procedure, which corresponds to the generation of Y. Functional time series $\mathbf{X}(s)^{(1)}, \cdots, \mathbf{X}(s)^{(N/2)}$ with a change point ($m = 1$) are generated as described above, and labeled as $(Y^{(1)}, \cdots, Y^{(N/2)}) = \mathbf{1}_{N/2}$. Subsequently, the process is repeated another $N/2$ times with $m = 0$ to generate sequences $\mathbf{X}(s)^{(1)}, \cdots, \mathbf{X}(s)^{(N/2)}$ without changes, labeled as $(Y^{(N/2+1)}, \cdots, Y^{(N)}) = \mathbf{0}_{N/2}$. These datasets, comprising data points with and without changes $(\mathbf{X}(s)^{(i)}, Y^{(i)})$ for $i \in [N]$, are combined and randomly arranged.

Finally, following the procedure described above, we can generate $(\mathbf{X}(s), Y)$ under three different scenarios for various combinations of $(T, N_{\text{train}}, N_{\text{test}})$, given different parameter settings (c, ξ). To enhance the diversity of our training samples and thereby bolster the robustness of our training process, we individually set the parameters (c, ξ) and the change point position in different experiments. We will provide details of our parameter configurations at the relevant sections of the experimental results. It is essential to note that although the test data may include changes with these parameters, it is sourced from the same distribution as the training set.

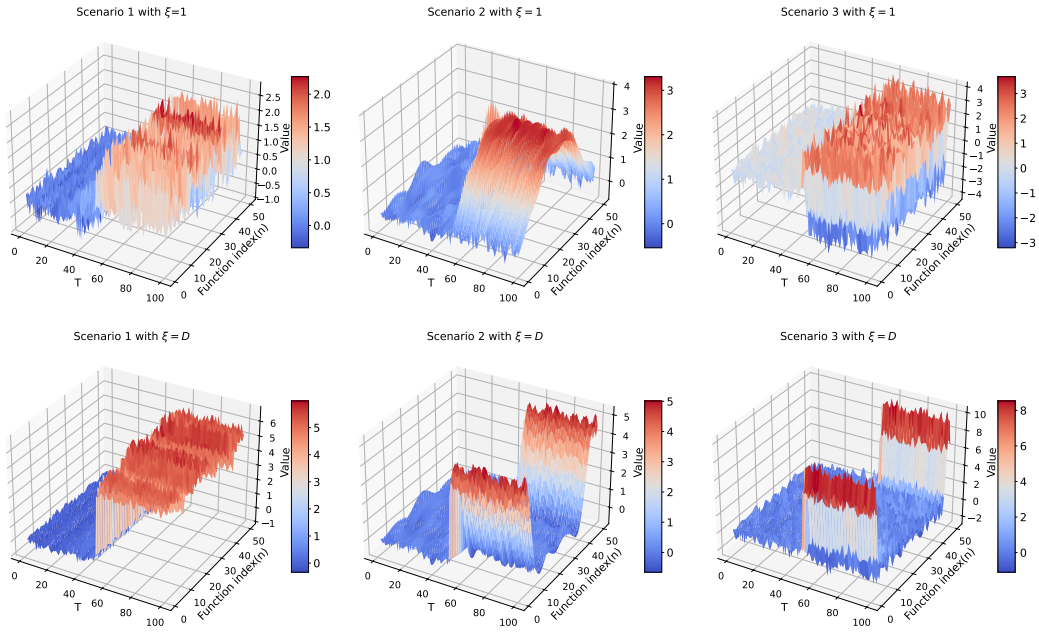


Figure 5: Difference between data generation scenarios with the varying number ξ of change functions. The first row corresponds to $\xi = 1$, while the second row represents $\xi = D$. From left to right, the three subplots depict Scenario 1, Scenario 2, and Scenario 3. The value of c is set to 1 in all six subplots.

Table 3: The classification accuracy of the FENet is tested across different scenarios, considering varying training sample sizes and time lengths. The training sample sizes N_{train} range from 200, 400, 600, 800, to 1000, while the time lengths T are set at 50 and 100.

Accuracy	N_{train}	200		400		600		800		1000	
Innovation	T	50	100	50	100	50	100	50	100	50	100
IID	S1	0.996	0.978	0.999	0.952	0.992	0.982	0.999	0.971	0.998	0.987
	S2	0.923	0.916	0.996	0.939	0.993	0.999	1	0.935	1	0.998
	S3	0.945	0.991	0.998	0.999	0.999	0.998	1	0.986	0.995	0.997
FAR	S1	0.956	0.866	0.993	0.961	0.998	0.979	0.995	0.985	0.998	0.984
	S2	0.961	0.929	0.991	0.992	0.993	0.997	0.997	0.985	1	0.995
	S3	0.984	0.972	0.996	0.95	0.991	0.983	0.997	0.999	0.998	0.995

C.2 FENet for Change-Point Detection

This subsection focuses on the accuracy of change-point detection, with particular emphasis on settings involving a single change point. We conduct five sets of experiments to evaluate the performance of the proposed method, including its accuracy under varying training sample sizes, comparisons with classical functional change-point detection methods, and robustness across complex test scenarios and different neural network architectures, as well as its computational scaling with the time length. Accuracy is defined as the proportion of correctly detected samples in the test set. Except when investigating the effect of network architecture by varying the number of layers or neurons, all networks are configured with two hidden layers containing 256 and 64 neurons, respectively. During training, the batch size is set to 64, the learning rate to 0.1, and the number of epochs to 100. The loss function used is binary cross entropy with logits loss, which is employed to approximate the non-differentiable 0–1 loss. and the Adam optimizer [Kingma and Ba, 2014] is employed.

Experiment 1: Test Accuracy under Different Training Sample Sizes We consider training sample sizes of $N_{\text{train}} = 200, 400, 600, 800$, and 1000, aiming to detect change points in data sequences of

Table 4: Comparison of test classification accuracy of different methods across diverse scenarios involving both IID and noise innovations. The methods considered are FFM (Fully Functional Method), TEV (Functional Principle Analysis with varying total explaining variance), and FENet. The number of basis functions with change point ξ is specified as 1 and D. The best results are highlighted in **bold**.

Innovation	IID						FAR					
Scenario	Scenario 1		Scenario 2		Scenario 3		Scenario 1		Scenario 2		Scenario 3	
Method	ξ=1	ξ=D	ξ=1	ξ=D	ξ=1	ξ=D	ξ=1	ξ=D	ξ=1	ξ=D	ξ=1	ξ=D
FFM	0.959	0.969	0.971	0.967	0.914	0.966	0.964	0.963	0.971	0.972	0.911	0.974
TEV-85	0.966	0.973	0.979	0.974	0.895	0.977	0.962	0.966	0.968	0.978	0.905	0.978
TEV-90	0.966	0.973	0.978	0.974	0.912	0.975	0.962	0.966	0.968	0.976	0.913	0.976
TEV-95	0.965	0.969	0.979	0.972	0.921	0.975	0.963	0.963	0.967	0.975	0.919	0.978
FENet	**0.973**	**0.981**	**0.989**	**0.978**	**0.99**	**0.987**	**0.97**	**0.979**	**0.98**	**0.991**	**0.982**	**0.986**

Table 5: Detection Accuracy with Varied Testing Sets and Fixed Training Set in Scenario 3 with IID Innovations. The training sample size ranges from 100 to 1000, specifically configured under Scenario 3 with IID innovation.

N_{train}		100	200	300	400	500	600	700	800	900	1000
IID	S1	0.918	0.924	0.91	0.923	0.922	0.903	0.915	0.907	0.904	0.925
	S2	0.91	0.974	0.911	0.944	0.928	0.972	0.949	0.99	0.948	0.974
	S3	0.974	0.991	0.994	0.994	0.995	0.999	1	0.996	1	1
FAR	S1	0.879	0.884	0.944	0.896	0.943	0.901	0.942	0.915	0.939	0.925
	S2	0.882	0.92	0.982	0.905	0.964	0.928	0.954	0.965	0.917	0.955
	S3	0.955	0.973	0.994	0.987	0.992	0.998	0.989	0.995	0.998	0.996

lengths $T = 50$ and $T = 100$. The computations are conducted on a test set comprising $N_{\text{test}} = 1000$ observations. For the training set, each sample is independently generated by drawing its SNR parameter c from a uniform distribution over $[0.1, 1]$, sampling the change-point position uniformly from $[0.1T, 0.9T]$, and randomly selecting the number of break functions ξ from $[1, D]$. For the test set, each sample is generated with c fixed at 1, the change point at $0.5T$, and ξ fixed at $D//2$.

Table 3 displays the test classification accuracy of the FENet across three scenarios, which involve two types of innovations. As shown in Table 3, the classification accuracy generally increases with larger N and decreases as T grows. This trend is fully consistent with the theoretical result established in Theorem 2, confirming that the model benefits from a larger number of observations per sample, while longer time series introduce greater complexity that slightly reduces accuracy. Moreover, the results indicate that the FENet performs robustly across both types of innovations, with only minor differences in accuracy, suggesting that the model effectively captures the underlying structure regardless of the innovation type.

Experiment 2: Comparison with Other Methods The fully functional method (FFM) [Aue et al., 2018] and functional principal component analysis (FPCA) with total variation explained (TVE) levels of 85%, 90%, and 95% [Berkes et al., 2009] are evaluated across multiple configurations. Three scenarios are considered using the same training data as in Experiment 1, with $N_{\text{train}} = 1000$ and $T = 100$. For testing, $N_{\text{test}} = 1000$ samples are generated for each $\xi \in \{1, D\}$. In all settings, the SNR parameter c is fixed at 1, and the change point is located at $0.5T$.

Table 4 presents a comparative analysis of the detection accuracy among our FENet method, the FFM and the TVE approach. The findings demonstrate the superior performance of our method in all cases, especially in Scenarios 3 with $\xi = 1$, where only one basis function changes among the 21 basis functions. Our FENet delivers outstanding results exceeding 98%, compared to other methods with accuracy around 91%.

Experiment 3: Robustness under Distributional Changes We conduct robustness experiments to examine how training a FENet under a single setting (Scenario 3 with IID innovations) generalizes to other configurations. The training sets are generated under S3–IID with $T = 50$ and varying sample sizes ($N_{\text{train}} = 100$–1000). For testing, datasets are separately generated for each of the three

scenarios (S1–S3) and both innovation types (IID and FAR), each containing 1000 samples. Other experimental settings are consistent with Experiment 1.

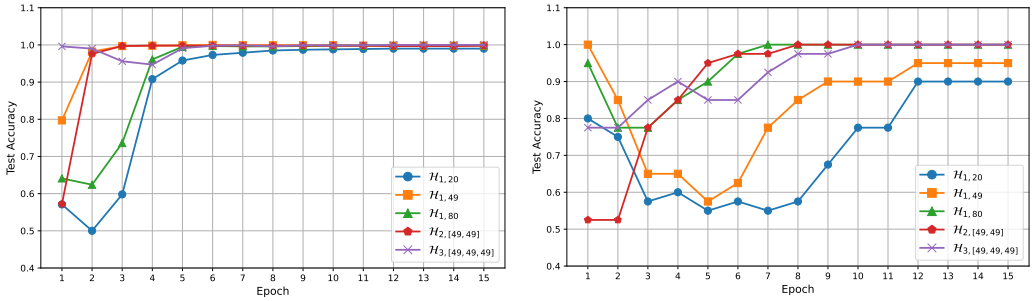


Figure 6: Test accuracy of FENets with different numbers of layers and nodes over training epochs under Scenario 3. The left and right panels (a and b) correspond to IID and FAR innovations, respectively. The training and test sets contain 500 and 1,000 samples, with time length $T = 50$. The network $\mathcal{H}_{1,49}$ represents the FCUSUM-based classifier in Theorem 1.

Except for one configuration in the third row of Table 5, the trained network is not aligned with the test data, allowing us to assess its robustness more systematically. The results show that our FENet performs best when the training and test settings coincide, yet it still achieves satisfactory accuracy under mismatched conditions, demonstrating the strong robustness of our method.

We further examine robustness to departures from the piecewise-constant mean and Gaussian innovation assumptions. Using the basis functions of Scenario 3, we train FENet (MLP) on piecewise-constant signals with IID Gaussian coefficient noise of standard deviation 0.20 and $c \sim \mathcal{U}[0.1, 1]$. We set $T = 100$, $N_{\text{train}} = 1000$, and $N_{\text{test}} = 1000$ per condition, with three seeds (101, 202, and 303). The trained models and hyperparameters are then fixed for all test conditions. The reference test setting retains the piecewise-constant Gaussian model with noise standard deviation 0.20 and $c = 1$.

We consider two additional test settings, both with coefficient-noise standard deviation 0.5 and signal strengths $c \in \{0.75, 0.5\}$. In the first, the mean changes linearly to a new persistent level over an interval of length $0.2T$ centered at $0.5T$, replacing the instantaneous jump with a smooth transition. In the second, Student-t_3 coefficient noise replaces Gaussian coefficient noise. Table 6 reports the mean accuracy and AUC, with standard deviations over three independent repetitions.

Table 6: Detection performance under model misspecification. Entries are means with standard deviations in parentheses over three repetitions. The reference uses coefficient-noise standard deviation 0.20, while the other settings use 0.5.

Test distribution	c	Accuracy	AUC
Piecewise-constant reference	1	0.9940 (0.0046)	1.0000 (0.0000)
Smooth persistent transition	0.75	0.9783 (0.0083)	0.9948 (0.0038)
Smooth persistent transition	0.5	0.9127 (0.0304)	0.9743 (0.0134)
Student-t_3 innovations	0.75	0.9687 (0.0074)	0.9837 (0.0013)
Student-t_3 innovations	0.5	0.9050 (0.0190)	0.9560 (0.0131)

Performance decreases as the signal weakens, but accuracy remains above 0.90 and AUC above 0.95 in both settings at $c = 0.5$. These results support robustness to the tested departures from the training model, including smooth transitions and heavy-tailed innovations under increased noise.

Experiment 4: Influence of Network Depth and Width on Detection To further examine the robustness of FENets under different settings, we evaluate FENets with varying depths and widths under two innovation types. As shown in the left panel of Figure 6, under IID innovations, both the FCUSUM-based classifier and alternative networks achieve high detection accuracy. The right panel of Figure 6 shows that under FAR innovations, deeper and wider networks outperform the FCUSUM-based classifier, indicating that FENets can represent the FCUSUM structure while retaining robustness in more complex settings.

Experiment 5: Computational Scaling with the Time Length The $T^3 \ln T$ term in Theorem 2 reflects the worst-case complexity of the theoretical network class and does not impose a necessary sample-size threshold. To assess how FENet scales with increasing T, we conduct experiments under Scenario 3 with $T \in \{50, 100, 200, 500\}$ and $N_{\text{train}} \in \{500, 1000\}$. For each configuration, we use three seeds (101, 202, and 303) and a common test set of 1,000 samples. For each seed, the 500-sample training set is a balanced subset of the corresponding 1,000-sample set. Table 7 reports the mean test accuracy and AUC, with standard deviations across the three seeds in parentheses.

Table 7: Test performance for increasing time length T and training sample size N_{train}. Entries are means with standard deviations in parentheses over three seeds.

T	N_{train}	Accuracy	AUC
50	500	0.8617 (0.0093)	0.9528 (0.0097)
50	1,000	0.8687 (0.0110)	0.9513 (0.0116)
100	500	0.9320 (0.0101)	0.9956 (0.0046)
100	1,000	0.9393 (0.0110)	0.9977 (0.0027)
200	500	0.9497 (0.0146)	0.9963 (0.0038)
200	1,000	0.9710 (0.0072)	1.0000 (0.0000)
500	500	0.9663 (0.0095)	0.9897 (0.0045)
500	1,000	0.9747 (0.0129)	0.9933 (0.0051)

We further measure computation time for the 256–64 MLP with Adam, batch size 64, and $N_{\text{train}} = 1000$. The deterministic embedding is computed once and cached. All timings are obtained on an 8-core Apple M2 CPU with 8 GB RAM. Table 8 reports means and standard deviations over the three seeds.

Table 8: Parameter counts and computation time for increasing T. Timing entries are means with standard deviations in parentheses over three seeds.

T	Input dimension	Parameters	Embedding(ms/sample)	One epoch(s)
50	2,450	643,969	0.027 (0.018)	0.035 (0.005)
100	9,900	2,551,169	0.030 (0.005)	0.121 (0.011)
200	39,800	10,205,569	0.103 (0.017)	0.428 (0.035)
500	249,500	63,888,769	1.991 (0.192)	3.346 (0.337)

The practical network remains computationally feasible at $T = 500$ and achieves high accuracy with 1,000 training samples. These results confirm that the sufficient $T^3 \ln T$ scaling in the worst-case VC bound should not be interpreted as an empirical minimum sample size for the fixed-width implementation.

C.3 FENet for Change-Point Localization

In this section, we present simulation results of change-point identification and localization by combining our trained FENet with Algorithm 1 (Appendix B.1).

We use the models trained under each scenario in Experiment 1 with $N_{\text{train}} = 1000$ and $T = 50$ as the classifiers in Algorithm 1. We then generate $N_{\text{test}} = 1000$ testing samples, each with a time length of T^*, containing either a single change point ($m = 1$) or two change points ($m = 2$). For the single change-point case ($m = 1$), the change point is located at $0.5T^*$. For the two change-point case ($m = 2$), the change points are positioned at $0.3T^*$ and $0.6T^*$. The procedure for generating multi-change-point functional time series follows Equation (16).

Before presenting the results, we briefly introduce the evaluation metrics used to assess the accuracy of change-point identification and localization. The metric P_m measures the accuracy of estimating the number of change points. To evaluate the accuracy of group membership and location estimation, we consider four complementary metrics: Normalized Mutual Information (NMI), Purity, Hausdorff Distance (HD), and Root Mean Square Error (RMSE). NMI and Purity quantify the agreement between the estimated and true change-point groupings, while Hausdorff Distance captures the worst-case discrepancy between the estimated and true sets of change-point locations. RMSE summarizes

Table 9: Localization for single and multiple change points under different scenarios with IID innovations. Three scenarios are established with different settings for the number of change points m and time length T^*. The configurations include one change point at 150 and 300, as well as two change points at 300.

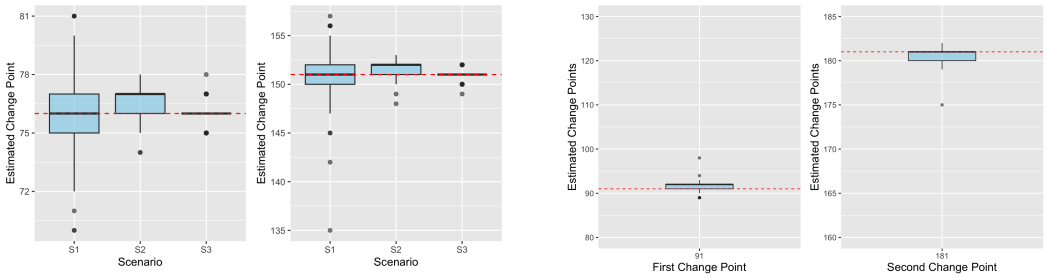
Scenario	(m, T^*)	Pm	NMI	Purity	HD	RMSE
S1	(1,150)	1.000	0.947	0.992	0.008	1.527
	(1,300)	0.987	0.968	0.996	0.007	1.605
	(2,300)	0.996	0.919	0.979	0.014	3.120
S2	(1,150)	1.000	0.967	0.996	0.004	0.892
	(1,300)	1.000	0.982	0.998	0.002	0.854
	(2,300)	1.000	0.928	0.983	0.010	2.636
S3	(1,150)	1.000	0.992	0.999	0.001	0.401
	(1,300)	1.000	0.995	0.999	0.001	0.437
	(2,300)	1.000	0.980	0.996	0.003	0.768

Table 10: Localization for single and multiple change points under different scenarios with FAR innovations. Three scenarios are established with different settings for the number of change points m and time length T^*. The configurations include one change point at 150 and 300, as well as two change points at 300.

Scenario	(m, T^*)	Pm	NMI	Purity	HD	RMSE
S1	(1,150)	1.000	0.940	0.991	0.009	1.500
	(1,300)	0.999	0.966	0.996	0.005	1.533
	(2,300)	0.998	0.953	0.989	0.007	1.881
S2	(1,150)	1.000	0.926	0.988	0.011	2.344
	(1,300)	0.999	0.959	0.995	0.005	2.080
	(2,300)	1.000	0.955	0.990	0.007	1.764
S3	(1,150)	1.000	0.929	0.990	0.010	1.628
	(1,300)	1.000	0.960	0.995	0.005	1.622
	(2,300)	1.000	0.938	0.986	0.009	2.321

the average squared deviation of estimated change-point positions from their true counterparts. Formal definitions and computational details are provided in Appendix B.2.

The experimental results under the three scenarios and two types of innovations are summarized in Table 9 and Table 10. Across all combinations of (m, T^*), the obtained values of Pm, NMI, and Purity are close to 1, while HD approaches 0, indicating that our algorithm performs well in both identifying the number of multiple change points and accurately locating their positions. The localization results are also detailed in three cases through boxplots, as shown in Figure 7. The first row of Figure 7 displays two subplots illustrating the distribution of change-point estimation in three scenarios with IID innovation for $(m, T^*) = (1, 150)$ and $(1, 300)$, where the change points are located at 76 and 151, respectively. The second row of the figure presents a subplot showcasing the distribution of change-point estimation in Scenario 2 with IID innovation for $(m, T^*) = (2, 300)$, featuring change points at 91 and 181. The accuracy of the estimations is notably high in both single and multiple change point cases.



(a) Single change point

(b) Two change points

Figure 7: Change-point localization under the IID innovation setting. Panel (a) shows localization results for a single change point at 76 when $T = 150$, and at 151 when $T = 300$. Panel (b) presents the two-change-point case with $m = 2$ and $T = 300$, where the first and second change points occur at 91 (left) and 181 (right), respectively. Red dashed lines indicate the true change-point positions.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Each claim in the abstract can be founded in the main paper.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: In the conclusion section, we have briefly discussed about future work, which address the limitation of the paper.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.

- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: The assumptions are listed in Section 3, and the detailed proofs are presented in Appendix A.2.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We describe the implementation details in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: The code required to replicate the simulation studies is publicly available. The code also contains an experiment based on another open-access data. The real-world server traffic data used in this paper are provided by the Shanghai Municipal Big Data Center and cannot be publicly released due to confidentiality and privacy restrictions. Since the main purpose of the simulations is to validate the proposed methodology under controlled settings, all data-generating mechanisms are explicitly specified and can be reproduced directly from the manuscript.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The training and testing details are described in Appendices B and C. The implementation details for the real-world experiments are provided in Section 4.

Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: The paper contains all the details about the numerical simulations we used.

   Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We provide the necessary information on the computational resources used in our experiments in Appendix B.

   Guidelines:

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

9. **Code of ethics**

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

Answer: [Yes]

Justification: This research has been conducted in strict accordance with the NeurIPS Code of Ethics.

Guidelines:

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

10. **Broader impacts**

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

Answer: [Yes]

Justification: We discuss the broader implications of FENet in the Introduction and Real Data Analysis section, highlighting its potential positive impact for detecting change points in functional time series.

Guidelines:

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

11. **Safeguards**

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

Answer: [N/A]

Justification: It is not relevant to our research topic.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: Yes. The real-world data used in this paper were provided by the Shanghai Big Data Center and are used under confidentiality restrictions. All baseline methods and related works are properly cited in the manuscript.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    Justification: No new public assets are introduced in this paper.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or human-subject studies.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or human-subject studies.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: We don't use LLMs to generate the methods.

    Guidelines:

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